

# CBERCOREAI COMPANY ASSIGNMENT

Data Collection: Gather historical data on stock prices, trading volumes, and other relevant technical indicators for the NYSE and LSE. This data will be sourced from reputable financial databases and APIs.

I have the relevant technical indicators such as moving averages, RSI, MACD etc ,calculate those from the available price data.I'll proceed with calculating some common technical indicators from the historical stock price data in the LSE and NYSE datasets, including:

1. Moving Averages (SMA & EMA)
2. Relative Strength Index (RSI)
3. Moving Average Convergence Divergence (MACD)
4. Bollinger Bands

```
import pandas as pd

# Load the datasets
lse_data = pd.read_csv('LSE Dataset.csv')
nyse_data = pd.read_csv('NYSE Dataset.csv')
```

```
# the primary few rows of each dataset
lse_data.head(), nyse_data.head()
```

	Date	Open	High	Low	Close	Adj
0	2001-07-20	392.255005	392.255005	392.255005	392.255005	275.853577
1	2001-07-23	370.760986	393.597992	365.388000	373.984985	263.005219
2	2001-07-24	374.523010	374.523010	356.252991	356.790985	250.913574
3	2001-07-25	349.268005	350.665009	333.148010	343.894989	241.844330
4	2001-07-26	348.192993	348.192993	340.670990	344.968994	242.599701

	ticker	date	open	high	low	close
0	A	1999-11-18	29.5594	32.4842	25.9889	28.5858
1	A	1999-11-19	27.8972	27.9371	25.8613	26.2311
2	A	1999-11-22	26.8370	28.5858	26.0278	28.5858
3	A	1999-11-23	27.6102	28.3377	25.9889	25.9889
4	A	1999-11-24	26.0637	27.2445	25.9889	26.6745)

The datasets contain relevant columns such as Date, Open, High, Low, Close, and Volume, which are important for calculating the technical indicators. I'll now proceed with the following:

1. Calculate Moving Averages (SMA & EMA).
2. Calculate the Relative Strength Index (RSI).
3. Calculate Moving Average Convergence Divergence (MACD).
4. Calculate Bollinger Bands.

```
import numpy as np

# Convert date columns to datetime format
lse_data['Date'] = pd.to_datetime(lse_data['Date'])
nyse_data['date'] = pd.to_datetime(nyse_data['date'])

# Function to calculate technical indicators
def calculate_indicators(df):
    df = df.sort_values(by='Date' if 'Date' in df.columns else 'date')

    # Moving Averages
    df['SMA_20'] = df['Close'].rolling(window=20).mean()
    df['EMA_20'] = df['Close'].ewm(span=20, adjust=False).mean()

    # RSI (Relative Strength Index)
    delta = df['Close'].diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
    rs = gain / loss
    df['RSI'] = 100 - (100 / (1 + rs))

    # MACD (Moving Average Convergence Divergence)
    df['EMA_12'] = df['Close'].ewm(span=12, adjust=False).mean()
    df['EMA_26'] = df['Close'].ewm(span=26, adjust=False).mean()
    df['MACD'] = df['EMA_12'] - df['EMA_26']
    df['Signal_Line'] = df['MACD'].ewm(span=9, adjust=False).mean()

    # Bollinger Bands
    df['Bollinger_Upper'] = df['SMA_20'] + (2 *
    df['Close'].rolling(window=20).std())
    df['Bollinger_Lower'] = df['SMA_20'] - (2 *
    df['Close'].rolling(window=20).std())

    return df

# Apply to both datasets
lse_data = calculate_indicators(lse_data)
nyse_data = calculate_indicators(nyse_data)

# Show the first few rows of the updated datasets
lse_data.head(), nyse_data.head()
```

```
-----  
-----  
KeyError                                Traceback (most recent call  
last)
```

```
File c:\Program Files\Python38\lib\site-packages\pandas\core\indexes\  
base.py:3653, in Index.get_loc(self, key)
```

```
    3652 try:  
-> 3653     return self._engine.get_loc(casted_key)  
    3654 except KeyError as err:
```

```
File c:\Program Files\Python38\lib\site-packages\pandas\_libs\  
index.pyx:147, in pandas._libs.index.IndexEngine.get_loc()
```

```
File c:\Program Files\Python38\lib\site-packages\pandas\_libs\  
index.pyx:176, in pandas._libs.index.IndexEngine.get_loc()
```

```
File pandas\_libs\hashtable_class_helper.pxi:7080, in  
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

```
File pandas\_libs\hashtable_class_helper.pxi:7088, in  
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

```
KeyError: 'Close'
```

The above exception was the direct cause of the following exception:

```
KeyError                                Traceback (most recent call  
last)
```

```
Cell In[3], line 36
```

```
    34 # Apply to both datasets  
    35 lse_data = calculate_indicators(lse_data)  
--> 36 nyse_data = calculate_indicators(nyse_data)  
    38 # Show the first few rows of the updated datasets  
    39 lse_data.head(), nyse_data.head()
```

```
Cell In[3], line 12, in calculate_indicators(df)
```

```
     9 df = df.sort_values(by='Date' if 'Date' in df.columns else  
'date')  
    11 # Moving Averages  
--> 12 df['SMA_20'] = df['Close'].rolling(window=20).mean()  
    13 df['EMA_20'] = df['Close'].ewm(span=20, adjust=False).mean()  
    15 # RSI (Relative Strength Index)
```

```
File c:\Program Files\Python38\lib\site-packages\pandas\core\  
frame.py:3761, in DataFrame.__getitem__(self, key)
```

```
    3759 if self.columns.nlevels > 1:  
    3760     return self._getitem_multilevel(key)  
-> 3761 indexer = self.columns.get_loc(key)  
    3762 if is_integer(indexer):  
    3763     indexer = [indexer]
```

```

File c:\Program Files\Python38\lib\site-packages\pandas\core\indexes\
base.py:3655, in Index.get_loc(self, key)
    3653     return self._engine.get_loc(casted_key)
    3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
    3656 except TypeError:
    3657     # If we have a listlike key, _check_indexing_error will
raise
    3658     # InvalidIndexError. Otherwise we fall through and re-
raise
    3659     # the TypeError.
    3660     self._check_indexing_error(key)

```

KeyError: 'Close'

The column name for stock prices in the NYSE dataset is lowercase ('close'), while in the LSE dataset it is capitalized ('Close'). I'll adjust the code to handle both cases and recalculate the technical indicators for the NYSE dataset.

The technical indicators have been successfully calculated for both datasets.

1. Moving Averages: SMA (Simple Moving Average) and EMA (Exponential Moving Average) over a 20-day period.
2. RSI (Relative Strength Index): Measures the speed and change of price movements to identify oversold conditions.
3. MACD (Moving Average Convergence Divergence): Shows the relationship between two EMAs (12 and 26-day periods).
4. Bollinger Bands: Upper and lower bands are set 2 standard deviations from the 20-day SMA.

```

# Adjust column name for consistency
nyse_data.rename(columns={'close': 'Close'}, inplace=True)

# Recalculate indicators for NYSE dataset
nyse_data = calculate_indicators(nyse_data)

```

```

# Show the first few rows of the updated datasets
lse_data.head(), nyse_data.head()

```

(	Date	Open	High	Low	Close	Adj
Close \						
0	2001-07-20	392.255005	392.255005	392.255005	392.255005	
275.853577						
1	2001-07-23	370.760986	393.597992	365.388000	373.984985	
263.005219						
2	2001-07-24	374.523010	374.523010	356.252991	356.790985	
250.913574						
3	2001-07-25	349.268005	350.665009	333.148010	343.894989	
241.844330						
4	2001-07-26	348.192993	348.192993	340.670990	344.968994	
242.599701						

Volume	SMA_20	EMA_20	RSI	EMA_12	EMA_26
--------	--------	--------	-----	--------	--------

MACD \						
0	584009.0	NaN	392.255005	NaN	392.255005	392.255005
0.000000						
1	3205437.0	NaN	390.515003	NaN	389.444233	390.901670 -
1.457437						
2	790321.0	NaN	387.303192	NaN	384.420656	388.374953 -
3.954297						
3	1381718.0	NaN	383.169077	NaN	378.185938	385.080141 -
6.894203						
4	1381718.0	NaN	379.530974	NaN	373.075639	382.108945 -
9.033306						

	Signal_Line	Bollinger_Upper	Bollinger_Lower			
0	0.000000	NaN	NaN			
1	-0.291487	NaN	NaN			
2	-1.024049	NaN	NaN			
3	-2.198080	NaN	NaN			
4	-3.565125	NaN	NaN			

  

ticker	date	open	high	low	Close	SMA_20
EMA_20 \						
0	A 1999-11-18	29.5594	32.4842	25.9889	28.5858	NaN
28.585800						
1	A 1999-11-19	27.8972	27.9371	25.8613	26.2311	NaN
28.361543						
2	A 1999-11-22	26.8370	28.5858	26.0278	28.5858	NaN
28.382901						
3	A 1999-11-23	27.6102	28.3377	25.9889	25.9889	NaN
28.154901						
4	A 1999-11-24	26.0637	27.2445	25.9889	26.6745	NaN
28.013910						

	RSI	EMA_12	EMA_26	MACD	Signal_Line	Bollinger_Upper
\						
0	NaN	28.585800	28.585800	0.000000	0.000000	NaN
1	NaN	28.223538	28.411378	-0.187839	-0.037568	NaN
2	NaN	28.279271	28.424298	-0.145027	-0.059060	NaN
3	NaN	27.926906	28.243898	-0.316992	-0.110646	NaN
4	NaN	27.734228	28.127646	-0.393418	-0.167200	NaN

	Bollinger_Lower
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN )

Data Preprocessing: Clean and preprocess the data by handling missing values, outliers, and performing necessary transformations to prepare the data for modeling.

1. Handle missing values: Fill or drop missing values, depending on the context.
2. Outlier detection: Identify and handle outliers using methods such as Z-score or IQR.
3. Feature scaling: Normalize or standardize the features if necessary.
4. Transformations: Convert dates into numerical values or extract useful features if needed.

```
# Check for missing values in both datasets
missing_lse = lse_data.isnull().sum()
missing_nyse = nyse_data.isnull().sum()

# Check for outliers in the numerical columns (using Z-score method)
from scipy.stats import zscore

# Calculate Z-scores for numeric columns to identify outliers
lse_data_numeric = lse_data.select_dtypes(include=np.number)
nyse_data_numeric = nyse_data.select_dtypes(include=np.number)

lse_zscores = np.abs(zscore(lse_data_numeric))
nyse_zscores = np.abs(zscore(nyse_data_numeric))

# Define a threshold for outliers (e.g., Z-score > 3)
outliers_lse = (lse_zscores > 3).sum(axis=0)
outliers_nyse = (nyse_zscores > 3).sum(axis=0)

missing_lse, missing_nyse, outliers_lse, outliers_nyse
```

(Date	0
Open	1
High	1
Low	1
Close	1
Adj Close	1
Volume	1
SMA_20	39
EMA_20	0
RSI	13
EMA_12	0
EMA_26	0
MACD	0
Signal_Line	0
Bollinger_Upper	39
Bollinger_Lower	39
dtype: int64,	
ticker	0
date	0
open	0
high	0
low	0

Close	0
SMA_20	19
EMA_20	0
RSI	13
EMA_12	0
EMA_26	0
MACD	0
Signal_Line	0
Bollinger_Upper	19
Bollinger_Lower	19
dtype: int64,	
Open	0
High	0
Low	0
Close	0
Adj Close	0
Volume	0
SMA_20	0
EMA_20	0
RSI	0
EMA_12	0
EMA_26	0
MACD	73
Signal_Line	67
Bollinger_Upper	0
Bollinger_Lower	0
dtype: int64,	
open	2
high	0
low	5
Close	3
SMA_20	0
EMA_20	0
RSI	0
EMA_12	0
EMA_26	0
MACD	144
Signal_Line	132
Bollinger_Upper	0
Bollinger_Lower	0
dtype: int64)	

Feature Engineering: Extract relevant features from the data that can help improve the model's predictive power. This may include technical indicators such as moving averages, relative strength index (RSI), and Bollinger Bands.

For feature engineering, I'll extract and prepare the relevant features from the data, focusing on the technical indicators I calculated:

1. Moving Averages (SMA and EMA)
2. RSI (Relative Strength Index)
3. MACD (Moving Average Convergence Divergence)
4. Bollinger Bands

These features will be used as inputs for the predictive model. I will focus on the following steps:

1. Extracting only the features that are most relevant for prediction.
2. Check that all features are aligned for both the LSE and NYSE datasets.
3. Creating a target variable, which may be the next day's stock price or price change.

```
# Feature extraction for model input
def extract_features(df):
    features = df[['Date', 'SMA_20', 'EMA_20', 'RSI', 'MACD',
                  'Signal_Line', 'Bollinger_Upper', 'Bollinger_Lower']]

    # Creating target variable: next day's Close price (next day's
    # prediction)
    features['Target'] = df['Close'].shift(-1)

    # Drop rows with NaN target (because the last row has no target)
    features = features.dropna()

    return features
```

```
# Extract features for both datasets
lse_features = extract_features(lse_data)
nyse_features = extract_features(nyse_data)
```

```
# Display the features
lse_features.head(), nyse_features.head()
```

C:\Users\hp\AppData\Local\Temp\ipykernel\_7208\3139717693.py:6:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.  
Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation:

[https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
features['Target'] = df['Close'].shift(-1)
```



```

-----
-----
KeyError                                Traceback (most recent call
last)
Cell In[8], line 15
    13 # Extract features for both datasets
    14 lse_features = extract_features(lse_data)
--> 15 nyse_features = extract_features(nyse_data)
    17 # Display the features
    18 lse_features.head(), nyse_features.head()

Cell In[8], line 3, in extract_features(df)
      2 def extract_features(df):
----> 3     features = df[['Date', 'SMA_20', 'EMA_20', 'RSI', 'MACD',
'Signal_Line', 'Bollinger_Upper', 'Bollinger_Lower']]
      5     # Creating target variable: next day's Close price (next
day's prediction)
      6     features['Target'] = df['Close'].shift(-1)

File c:\Program Files\Python38\lib\site-packages\pandas\core\
frame.py:3767, in DataFrame.__getitem__(self, key)
    3765     if is_iterator(key):
    3766         key = list(key)
-> 3767     indexer = self.columns._get_indexer_strict(key, "columns")
[1]
    3769 # take() does not accept boolean indexers
    3770 if getattr(indexer, "dtype", None) == bool:

File c:\Program Files\Python38\lib\site-packages\pandas\core\indexes\
base.py:5877, in Index._get_indexer_strict(self, key, axis_name)
    5874 else:
    5875     keyarr, indexer, new_indexer =
self._reindex_non_unique(keyarr)
-> 5877 self._raise_if_missing(keyarr, indexer, axis_name)
    5879 keyarr = self.take(indexer)
    5880 if isinstance(key, Index):
    5881     # GH 42790 - Preserve name from an Index

File c:\Program Files\Python38\lib\site-packages\pandas\core\indexes\
base.py:5941, in Index._raise_if_missing(self, key, indexer,
axis_name)
    5938     raise KeyError(f"None of [{key}] are in the
[{axis_name}]")
    5940 not_found = list(ensure_index(key)[missing_mask.nonzero()
[0]].unique())
-> 5941 raise KeyError(f"{not_found} not in index")

KeyError: "['Date'] not in index"

```

I noticed that the NYSE dataset has the date column in lowercase, while the LSE dataset has the Date column with an uppercase "D". I'll adjust for this discrepancy in the column names.

```
# Standardize date column names to 'Date' for both datasets
lse_data.rename(columns={'Date': 'Date'}, inplace=True)
nyse_data.rename(columns={'date': 'Date'}, inplace=True)

# Feature extraction
def extract_features(df):
    features = df[['Date', 'SMA_20', 'EMA_20', 'RSI', 'MACD',
'Signal_Line', 'Bollinger_Upper', 'Bollinger_Lower']]

    # Creating target variable: next day's Close price (next day's
prediction)
    features['Target'] = df['Close'].shift(-1)

    # Drop rows with NaN target (because the last row has no target)
    features = features.dropna()

    return features

# Extract features for both datasets
lse_features = extract_features(lse_data)
nyse_features = extract_features(nyse_data)
```

```
# Display the features
lse_features.head(), nyse_features.head()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_7208\3004928171.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
features['Target'] = df['Close'].shift(-1)
C:\Users\hp\AppData\Local\Temp\ipykernel_7208\3004928171.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
features['Target'] = df['Close'].shift(-1)

(
      Date      SMA_20      EMA_20      RSI      MACD
Signal_Line \
19 2001-08-16  377.115401  384.698800  76.991105  2.802785  -
0.011939
20 2001-08-17  377.357201  385.879010  76.576404  3.392145
```

```

0.668877
21 2001-08-20 378.270702 386.486248 69.564999 3.429459
1.220994
22 2001-08-21 380.312552 387.547366 70.338762 3.848226
1.746440
23 2001-08-22 382.999202 388.507426 64.285293 4.132465
2.223645

```

	Bollinger_Upper	Bollinger_Lower	Target			
19	409.820331	344.410471	397.091003			
20	410.600504	344.113897	392.255005			
21	412.122359	344.419044	397.627991			
22	413.631215	346.993889	397.627991			
23	412.387195	353.611209	395.479004	,		

  

	Date	SMA_20	EMA_20	RSI	MACD	Signal_Line
\						
19	1999-12-16	28.283215	28.659844	67.898652	0.241679	0.142963
20	1999-12-17	28.346240	28.772840	61.039735	0.304534	0.175277
21	1999-12-20	28.557550	28.933265	62.897012	0.399050	0.220032
22	1999-12-21	28.642655	29.062278	60.400407	0.455040	0.267033
23	1999-12-22	28.888045	29.236985	59.843124	0.542286	0.322084

	Bollinger_Upper	Bollinger_Lower	Target
19	30.924960	25.641470	29.8463
20	31.077024	25.615456	30.4573
21	31.253017	25.862083	30.2879
22	31.447154	25.838156	30.8967
23	31.571082	26.205008	32.3168 )

Model Selection: Choose a suitable machine learning algorithm for this problem, such as a recurrent neural network (RNN), long short-term memory (LSTM) network, or a gradient boosting model.

Normalizes the stock data to a range between 0 and 1 using MinMaxScaler, which helps the LSTM model perform better. Then, it creates sequences of past stock data to predict future stock prices. These input features are paired with the target values (next day's stock price). This prepares the data in a format which is suitable for training an LSTM model, allowing it to learn patterns in the stock data over time. The final result is a dataset ready for model training with input sequences (X) and their corresponding targets (y).

```

from sklearn.preprocessing import MinMaxScaler
import numpy as np

# Normalize the data using MinMaxScaler (since LSTM models benefit
from scaled data)
scaler = MinMaxScaler(feature_range=(0, 1))

# Normalize features for LSE and NYSE datasets

```

```

lse_features_scaled = scaler.fit_transform(lse_features[['SMA_20',
'EMA_20', 'RSI', 'MACD', 'Signal_Line', 'Bollinger_Upper',
'Bollinger_Lower']])
nyse_features_scaled = scaler.fit_transform(nyse_features[['SMA_20',
'EMA_20', 'RSI', 'MACD', 'Signal_Line', 'Bollinger_Upper',
'Bollinger_Lower']])

# Prepare data for LSTM (create sequences)
def create_sequences(features_scaled, target, sequence_length=60):
    X, y = [], []
    for i in range(sequence_length, len(features_scaled)):
        X.append(features_scaled[i-sequence_length:i])
        y.append(target[i])
    return np.array(X), np.array(y)

# For LSE and NYSE, create sequences for input and output
lse_X, lse_y = create_sequences(lse_features_scaled,
lse_features['Target'].values)
nyse_X, nyse_y = create_sequences(nyse_features_scaled,
nyse_features['Target'].values)

# Check the shape of the prepared data
lse_X.shape, lse_y.shape, nyse_X.shape, nyse_y.shape

((4324, 60, 7), (4324,), (6217, 60, 7), (6217,))

```

Model Training: Train the selected model using the preprocessed data, with a suitable split between training and validation sets.

The data is split into 80% training and 20% validation sets to evaluation of the model's performance.

An LSTM model is built with two LSTM layers and dropout layers to prevent overfitting.

The model is compiled using the Adam optimizer and mean squared error (MSE) loss function for regression tasks.

The model is trained for 20 epochs with a batch size of 32, using the training data and validating on a separate validation set.

```

from sklearn.model_selection import train_test_split

# Split the data into training and validation sets (80% train, 20%
validation)
lse_X_train, lse_X_val, lse_y_train, lse_y_val =
train_test_split(lse_X, lse_y, test_size=0.2, shuffle=False)
nyse_X_train, nyse_X_val, nyse_y_train, nyse_y_val =
train_test_split(nyse_X, nyse_y, test_size=0.2, shuffle=False)

# Build LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout

```

```

model = Sequential()

# Add LSTM layers
model.add(LSTM(units=50, return_sequences=True,
input_shape=(lse_X_train.shape[1], lse_X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))

# Output layer
model.add(Dense(units=1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
history = model.fit(lse_X_train, lse_y_train, epochs=20,
batch_size=32, validation_data=(lse_X_val, lse_y_val))

Epoch 1/20
109/109 [=====] - 17s 84ms/step - loss:
1082454.3750 - val_loss: 11602088.0000
Epoch 2/20
109/109 [=====] - 7s 64ms/step - loss:
1068537.2500 - val_loss: 11562047.0000
Epoch 3/20
109/109 [=====] - 8s 71ms/step - loss:
1058200.0000 - val_loss: 11523916.0000
Epoch 4/20
109/109 [=====] - 7s 61ms/step - loss:
1048282.3750 - val_loss: 11486574.0000
Epoch 5/20
109/109 [=====] - 7s 61ms/step - loss:
1038449.5000 - val_loss: 11449679.0000
Epoch 6/20
109/109 [=====] - 7s 61ms/step - loss:
1028721.5000 - val_loss: 11413363.0000
Epoch 7/20
109/109 [=====] - 7s 61ms/step - loss:
1019489.6250 - val_loss: 11377503.0000
Epoch 8/20
109/109 [=====] - 7s 61ms/step - loss:
1010113.5625 - val_loss: 11341700.0000
Epoch 9/20
109/109 [=====] - 7s 67ms/step - loss:
1001010.7500 - val_loss: 11306035.0000
Epoch 10/20
109/109 [=====] - 9s 79ms/step - loss:
991786.1875 - val_loss: 11270416.0000

```

```

Epoch 11/20
109/109 [=====] - 7s 64ms/step - loss:
983102.6875 - val_loss: 11235259.0000
Epoch 12/20
109/109 [=====] - 7s 68ms/step - loss:
974064.3125 - val_loss: 11200347.0000
Epoch 13/20
109/109 [=====] - 7s 62ms/step - loss:
965216.3750 - val_loss: 11165496.0000
Epoch 14/20
109/109 [=====] - 7s 62ms/step - loss:
956072.4375 - val_loss: 11130547.0000
Epoch 15/20
109/109 [=====] - 7s 61ms/step - loss:
947235.6250 - val_loss: 11095877.0000
Epoch 16/20
109/109 [=====] - 7s 64ms/step - loss:
938674.9375 - val_loss: 11061472.0000
Epoch 17/20
109/109 [=====] - 7s 63ms/step - loss:
930457.0000 - val_loss: 11026930.0000
Epoch 18/20
109/109 [=====] - 7s 65ms/step - loss:
921660.5625 - val_loss: 10992560.0000
Epoch 19/20
109/109 [=====] - 7s 62ms/step - loss:
912967.6875 - val_loss: 10958496.0000
Epoch 20/20
109/109 [=====] - 7s 65ms/step - loss:
905319.3750 - val_loss: 10924581.0000

```

Model Evaluation: Evaluate the performance of the trained model using metrics such as mean absolute error (MAE), mean squared error (MSE), and R-squared.

1. Mean Absolute Error (MAE): Measures the average magnitude of errors in the model's predictions, giving an idea of how far off the predictions are, on average.
2. Mean Squared Error (MSE): Squares the errors to penalize larger errors more heavily, providing a measure of how well the model is fitting the data.
3. R-Squared ( $R^2$ ): Measures the proportion of variance in the data that the model explains, with a higher value indicating better performance.

```

from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

# Predict the stock prices using the trained model on the validation
data
lse_y_pred = model.predict(lse_X_val)

# Evaluate the model using MAE, MSE, and  $R^2$ 
mae = mean_absolute_error(lse_y_val, lse_y_pred)

```

```

mse = mean_squared_error(lse_y_val, lse_y_pred)
r2 = r2_score(lse_y_val, lse_y_pred)

# Display the evaluation metrics
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-Squared (R²): {r2}")

28/28 [=====] - 3s 20ms/step
Mean Absolute Error (MAE): 3224.9298897532217
Mean Squared Error (MSE): 10924581.53104432
R-Squared (R²): -19.832188146963134

```

Hyperparameter Tuning: Perform hyperparameter tuning to optimize the model's performance, using techniques such as grid search, random search, or Bayesian optimization.

Hyperparameter tuning for an LSTM model using RandomizedSearchCV from Scikit-learn, defines a build\_model function that constructs an LSTM model with adjustable parameters, including the number of LSTM units, dropout rate, batch size, number of epochs, and optimizer type. By evaluating different combinations of hyperparameters, it identifies the best set that improves the model's performance. Finally, the code outputs the best hyperparameters, allowing to optimize the model for better predictive accuracy on the stock price data.

```

pip install scikeras

```

```

Requirement already satisfied: scikeras in c:\program files\python38\
lib\site-packages (0.12.0)Note: you may need to restart the kernel to
use updated packages.

```

```

Requirement already satisfied: packaging>=0.21 in c:\program files\
python38\lib\site-packages (from scikeras) (23.2)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\program
files\python38\lib\site-packages (from scikeras) (1.3.2)
Requirement already satisfied: tensorflow-io-gcs-
filesystem<0.32,>=0.23.1 in c:\program files\python38\lib\site-
packages (from scikeras) (0.28.0)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\program files\
python38\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(1.23.1)
Requirement already satisfied: scipy>=1.5.0 in c:\program files\
python38\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\program files\
python38\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\program
files\python38\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(3.4.0)

```

```

from scikeras.wrappers import KerasRegressor
from sklearn.model_selection import RandomizedSearchCV
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout

```

```

import numpy as np

# Function to build the Keras model
def build_model(units=50, dropout_rate=0.2, optimizer='adam'):
    model = Sequential()
    model.add(LSTM(units=units, return_sequences=True,
input_shape=(lse_X_train.shape[1], lse_X_train.shape[2])))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units, return_sequences=False))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units=1))
    model.compile(optimizer=optimizer, loss='mean_squared_error')
    return model

# Define the parameter grid for Random Search
param_dist = {
    'model__units': [50, 100, 150],
    'model__dropout_rate': [0.2, 0.3, 0.4],
    'batch_size': [16, 32, 64],
    'epochs': [10, 20, 30],
    'optimizer': ['adam', 'rmsprop']
}

# Wrap the Keras model using scikeras KerasRegressor
model = KerasRegressor(model=build_model, verbose=0)

# Perform Random Search with 3-fold cross-validation
random_search = RandomizedSearchCV(estimator=model,
param_distributions=param_dist, n_iter=10, cv=3, verbose=2)

# Assuming lse_X_train and lse_y_train are defined
random_search_result = random_search.fit(lse_X_train, lse_y_train)

# Display the best hyperparameters
print(f"Best Hyperparameters: {random_search_result.best_params_}")

Fitting 3 folds for each of 10 candidates, totalling 30 fits
[CV] END batch_size=32, epochs=20, model__dropout_rate=0.2,
model__units=50, optimizer=adam; total time= 1.8min
[CV] END batch_size=32, epochs=20, model__dropout_rate=0.2,
model__units=50, optimizer=adam; total time= 1.6min
[CV] END batch_size=32, epochs=20, model__dropout_rate=0.2,
model__units=50, optimizer=adam; total time= 1.6min
[CV] END batch_size=64, epochs=20, model__dropout_rate=0.3,
model__units=150, optimizer=adam; total time= 8.7min
[CV] END batch_size=64, epochs=20, model__dropout_rate=0.3,
model__units=150, optimizer=adam; total time= 8.6min
[CV] END batch_size=64, epochs=20, model__dropout_rate=0.3,
model__units=150, optimizer=adam; total time=10.3min
[CV] END batch_size=64, epochs=30, model__dropout_rate=0.2,

```



```
model__units=100, optimizer=adam; total time= 4.7min
[CV] END batch_size=64, epochs=30, model__dropout_rate=0.2,
model__units=100, optimizer=adam; total time= 2.3min
[CV] END batch_size=64, epochs=30, model__dropout_rate=0.2,
model__units=100, optimizer=adam; total time= 3.9min
[CV] END batch_size=16, epochs=20, model__dropout_rate=0.4,
model__units=150, optimizer=rmsprop; total time= 7.6min
[CV] END batch_size=16, epochs=20, model__dropout_rate=0.4,
model__units=150, optimizer=rmsprop; total time= 5.6min
[CV] END batch_size=16, epochs=20, model__dropout_rate=0.4,
model__units=150, optimizer=rmsprop; total time=722.8min
[CV] END batch_size=32, epochs=20, model__dropout_rate=0.4,
model__units=50, optimizer=adam; total time= 1.9min
[CV] END batch_size=32, epochs=20, model__dropout_rate=0.4,
model__units=50, optimizer=adam; total time= 1.7min
[CV] END batch_size=32, epochs=20, model__dropout_rate=0.4,
model__units=50, optimizer=adam; total time= 2.3min
[CV] END batch_size=32, epochs=30, model__dropout_rate=0.3,
model__units=150, optimizer=adam; total time=54.5min
[CV] END batch_size=32, epochs=30, model__dropout_rate=0.3,
model__units=150, optimizer=adam; total time=1062.3min
[CV] END batch_size=32, epochs=30, model__dropout_rate=0.3,
model__units=150, optimizer=adam; total time=20.1min
[CV] END batch_size=16, epochs=10, model__dropout_rate=0.3,
model__units=100, optimizer=rmsprop; total time= 3.8min
[CV] END batch_size=16, epochs=10, model__dropout_rate=0.3,
model__units=100, optimizer=rmsprop; total time= 3.5min
[CV] END batch_size=16, epochs=10, model__dropout_rate=0.3,
model__units=100, optimizer=rmsprop; total time= 4.7min
[CV] END batch_size=64, epochs=20, model__dropout_rate=0.3,
model__units=50, optimizer=adam; total time= 1.5min
[CV] END batch_size=64, epochs=20, model__dropout_rate=0.3,
model__units=50, optimizer=adam; total time= 1.7min
[CV] END batch_size=64, epochs=20, model__dropout_rate=0.3,
model__units=50, optimizer=adam; total time= 1.6min
[CV] END batch_size=32, epochs=30, model__dropout_rate=0.4,
model__units=100, optimizer=adam; total time= 5.5min
[CV] END batch_size=32, epochs=30, model__dropout_rate=0.4,
model__units=100, optimizer=adam; total time= 7.7min
[CV] END batch_size=32, epochs=30, model__dropout_rate=0.4,
model__units=100, optimizer=adam; total time= 7.6min
[CV] END batch_size=64, epochs=30, model__dropout_rate=0.3,
model__units=100, optimizer=rmsprop; total time= 3.5min
[CV] END batch_size=64, epochs=30, model__dropout_rate=0.3,
model__units=100, optimizer=rmsprop; total time= 5.6min
[CV] END batch_size=64, epochs=30, model__dropout_rate=0.3,
model__units=100, optimizer=rmsprop; total time= 5.1min
Best Hyperparameters: {'optimizer': 'rmsprop', 'model__units': 150,
'model__dropout_rate': 0.4, 'epochs': 20, 'batch_size': 16}
```

```
pip install flask tensorflow scikit-learn numpy pandas
```

Requirement already satisfied: flask in c:\program files\python38\lib\site-packages (3.0.3)

Requirement already satisfied: tensorflow in c:\program files\python38\lib\site-packages (2.11.1)

Requirement already satisfied: scikit-learn in c:\program files\python38\lib\site-packages (1.3.2)

Requirement already satisfied: numpy in c:\program files\python38\lib\site-packages (1.23.1)

Requirement already satisfied: pandas in c:\program files\python38\lib\site-packages (2.0.3)

Requirement already satisfied: Werkzeug>=3.0.0 in c:\program files\python38\lib\site-packages (from flask) (3.0.2)

Requirement already satisfied: Jinja2>=3.1.2 in c:\program files\python38\lib\site-packages (from flask) (3.1.2)

Requirement already satisfied: itsdangerous>=2.1.2 in c:\program files\python38\lib\site-packages (from flask) (2.1.2)

Requirement already satisfied: click>=8.1.3 in c:\program files\python38\lib\site-packages (from flask) (8.1.7)

Requirement already satisfied: blinker>=1.6.2 in c:\program files\python38\lib\site-packages (from flask) (1.7.0)

Requirement already satisfied: importlib-metadata>=3.6.0 in c:\program files\python38\lib\site-packages (from flask) (6.8.0)

Requirement already satisfied: tensorflow-intel==2.11.1 in c:\program files\python38\lib\site-packages (from tensorflow) (2.11.1)

Requirement already satisfied: absl-py>=1.0.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (1.4.0)

Requirement already satisfied: astunparse>=1.6.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (1.6.3)

Requirement already satisfied: flatbuffers>=2.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (24.3.25)

Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (0.4.0)

Requirement already satisfied: google-pasta>=0.1.1 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (0.2.0)

Requirement already satisfied: h5py>=2.9.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (3.11.0)

Requirement already satisfied: libclang>=13.0.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (18.1.1)

Requirement already satisfied: opt-einsum>=2.3.2 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (3.3.0)

Requirement already satisfied: packaging in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (23.2)

Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (3.19.6)

Requirement already satisfied: setuptools in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (75.3.0)

Requirement already satisfied: six>=1.12.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (1.16.0)

Requirement already satisfied: termcolor>=1.1.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (2.4.0)

Requirement already satisfied: typing-extensions>=3.6.6 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (4.11.0)

Requirement already satisfied: wrapt>=1.11.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (1.16.0)

Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (1.64.0)

Requirement already satisfied: tensorboard<2.12,>=2.11 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (2.11.2)

Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (2.11.0)

Requirement already satisfied: keras<2.12,>=2.11.0 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (2.11.0)

Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\program files\python38\lib\site-packages (from tensorflow-intel==2.11.1->tensorflow) (0.28.0)

Requirement already satisfied: scipy>=1.5.0 in c:\program files\python38\lib\site-packages (from scikit-learn) (1.10.1)

Requirement already satisfied: joblib>=1.1.1 in c:\program files\python38\lib\site-packages (from scikit-learn) (1.4.0)

Requirement already satisfied: threadpoolctl>=2.0.0 in c:\program files\python38\lib\site-packages (from scikit-learn) (3.4.0)

Requirement already satisfied: python-dateutil>=2.8.2 in c:\program files\python38\lib\site-packages (from pandas) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in c:\program files\python38\lib\site-packages (from pandas) (2023.3.post1)

Requirement already satisfied: tzdata>=2022.1 in c:\program files\python38\lib\site-packages (from pandas) (2023.3)

Requirement already satisfied: colorama in c:\program files\python38\lib\site-packages (from click>=8.1.3->flask) (0.4.6)

Requirement already satisfied: zipp>=0.5 in c:\program files\python38\lib\site-packages (from importlib-metadata>=3.6.0->flask) (3.17.0)

Requirement already satisfied: MarkupSafe>=2.0 in c:\program files\python38\lib\site-packages (from Jinja2>=3.1.2->flask) (2.1.3)

Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\program files\python38\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.11.1->tensorflow) (0.44.0)

Requirement already satisfied: google-auth<3,>=1.6.3 in c:\program files\python38\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (2.29.0)

Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\program files\python38\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (0.4.6)

Requirement already satisfied: markdown>=2.6.8 in c:\program files\python38\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (3.6)

Requirement already satisfied: requests<3,>=2.21.0 in c:\program files\python38\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (2.32.3)

Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\program files\python38\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (0.6.1)

Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\program files\python38\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (1.8.1)

Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\program files\python38\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (5.3.2)

Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\program files\python38\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (0.4.0)

Requirement already satisfied: rsa<5,>=3.1.4 in c:\program files\python38\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (4.9)

Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\program files\python38\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (2.0.0)

Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hp\appdata\roaming\python\python38\site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (3.3.0)

Requirement already satisfied: idna<4,>=2.5 in c:\program files\python38\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1->tensorflow) (2.10)

Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hp\appdata\roaming\python\python38\site-packages (from

```
requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.1->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\
appdata\roaming\python\python38\site-packages (from
requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.1->tensorflow) (2023.7.22)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in c:\program
files\python38\lib\site-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.1-
>tensorflow) (0.6.0)
Requirement already satisfied: oauthlib>=3.0.0 in c:\program files\
python38\lib\site-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.1->tensorflow) (3.2.2)
Note: you may need to restart the kernel to use updated packages.
```

Model Deployment: Deploy the final model in a production-ready environment, where it can be used to generate predictions on new, unseen data.

```
# Save the best model after RandomizedSearchCV
best_model = random_search_result.best_estimator_.model_
best_model.save("./best_lstm_model.h5") # Save the model in the
current working directory

import os
print(os.getcwd()) # Checking current working directory
print(os.listdir(".")) # List files in the directory

c:\Users\hp\OneDrive\Desktop\CbercoreIT Company
['app.py', 'best_lstm_model.h5', 'index.ipynb', 'LSE Dataset.csv',
'NYSE Dataset.csv']

from keras.models import load_model

# Load the saved model
loaded_model = load_model("./best_lstm_model.h5")

# Predict using the loaded model
predictions = loaded_model.predict(lse_X_val)

28/28 [=====] - 11s 84ms/step

print(type(best_model))

<class 'keras.engine.sequential.Sequential'>
```

Github Link of this assignment

<https://github.com/nidhisahani-glitch/Cbercore-IT-Assignment.git>

Deployment Link

[cbercoreitassignment-8s0vkx839-nidhi-sahanis-projects.vercel.app](https://cbercoreitassignment-8s0vkx839-nidhi-sahanis-projects.vercel.app)

Domain Link

[cbercoreitassignment.vercel.app](https://cbercoreitassignment.vercel.app)

Deployment Dashboard Link

[https://vercel.com/nidhi-sahanis-projects/cbercoreit\\_assignment](https://vercel.com/nidhi-sahanis-projects/cbercoreit_assignment)