**Statistical Learning Lab**

Assignment - 8

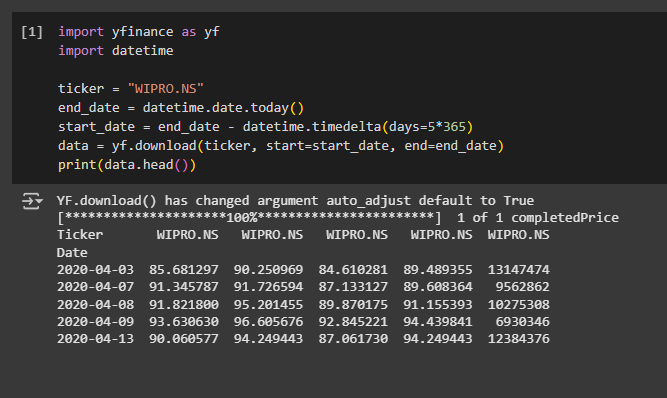
**Recurrent Neural Network for Stock Prediction**

Anurag Singha | 22IM30007 

1. Choose a stock of your choice from NIFTY 50 list from Yahoo Finance.

Choosen **WIPRO** Stock for this

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.

from sklearn.preprocessing import MinMaxScaler

import numpy as np

data = data[['Close']]

dataset = data.values

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(dataset)

test\_data\_len = 90

train\_data = scaled\_data[:-test\_data\_len]

test\_data = scaled\_data[-test\_data\_len:]

4. Use a predictive model using 3 LSTM layers, with past 60 days data, ntimestep = 60, dropout regularization ndrop = 0.2.

from keras.models import Sequential

from keras.layers import LSTM, Dense, Dropout

def create\_dataset(data, timestep):

    x, y = [], []

    for i in range(timestep, len(data)):

        x.append(data[i - timestep:i, 0])

        y.append(data[i, 0])

    return np.array(x), np.array(y)

x\_train, y\_train = create\_dataset(train\_data, 60)

x\_test, y\_test = create\_dataset(test\_data, 60)

x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1))

model = Sequential()

model.add(LSTM(50, return\_sequences=True, input\_shape=(x\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(50, return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(50))

model.add(Dropout(0.2))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(x\_train, y\_train, epochs=20, batch\_size=32)

**34/34** ━━━━━━━━━━━━━━━━━━━━ **10s** 84ms/step - loss: 0.1126

Epoch 2/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **3s** 79ms/step - loss: 0.0091

Epoch 3/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **3s** 79ms/step - loss: 0.0063

Epoch 4/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **6s** 109ms/step - loss: 0.0055

Epoch 5/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **4s** 81ms/step - loss: 0.0053

Epoch 6/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **4s** 125ms/step - loss: 0.0047

Epoch 7/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **5s** 112ms/step - loss: 0.0044

Epoch 8/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **4s** 82ms/step - loss: 0.0047

Epoch 9/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **6s** 112ms/step - loss: 0.0050

Epoch 10/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **4s** 82ms/step - loss: 0.0044

Epoch 11/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **3s** 79ms/step - loss: 0.0038

Epoch 12/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **6s** 117ms/step - loss: 0.0040

Epoch 13/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **3s** 78ms/step - loss: 0.0040

Epoch 14/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **5s** 84ms/step - loss: 0.0035

Epoch 15/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **6s** 116ms/step - loss: 0.0043

Epoch 16/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **4s** 79ms/step - loss: 0.0036

Epoch 17/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **3s** 78ms/step - loss: 0.0035

Epoch 18/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **7s** 123ms/step - loss: 0.0029

Epoch 19/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **4s** 82ms/step - loss: 0.0038

Epoch 20/20

**34/34** ━━━━━━━━━━━━━━━━━━━━ **3s** 84ms/step - loss: 0.0036

<keras.src.callbacks.history.History at 0x7ff72f34be10>

5. Create the plots comparing observed value of the test data and the predictive value.

forecast = model.predict(x\_test)

forecast = scaler.inverse\_transform(forecast)

observed = scaler.inverse\_transform(y\_test.reshape(-1, 1))

import matplotlib.pyplot as plt

plt.figure(figsize=(14, 5))

plt.plot(observed, label='Observed Price')

plt.plot(forecast, label='Forecasted Price')

plt.title('WIPRO Stock Price Forecast')

plt.xlabel('Time')

plt.ylabel('Price')

plt.legend()

plt.savefig("stock\_forecast\_plot.png")

plt.show()

residuals = observed - forecast

plt.figure(figsize=(14, 5))

plt.plot(residuals, label='Forecast Error (Residual)')

plt.axhline(y=0, color='r', linestyle='--')

plt.title('Residual Plot: Observed - Forecasted')

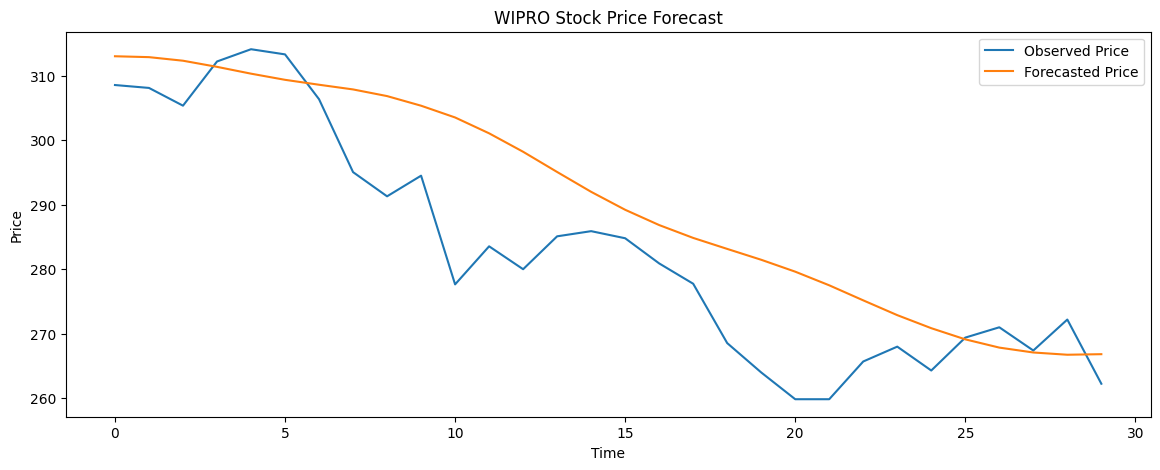
plt.xlabel('Time')

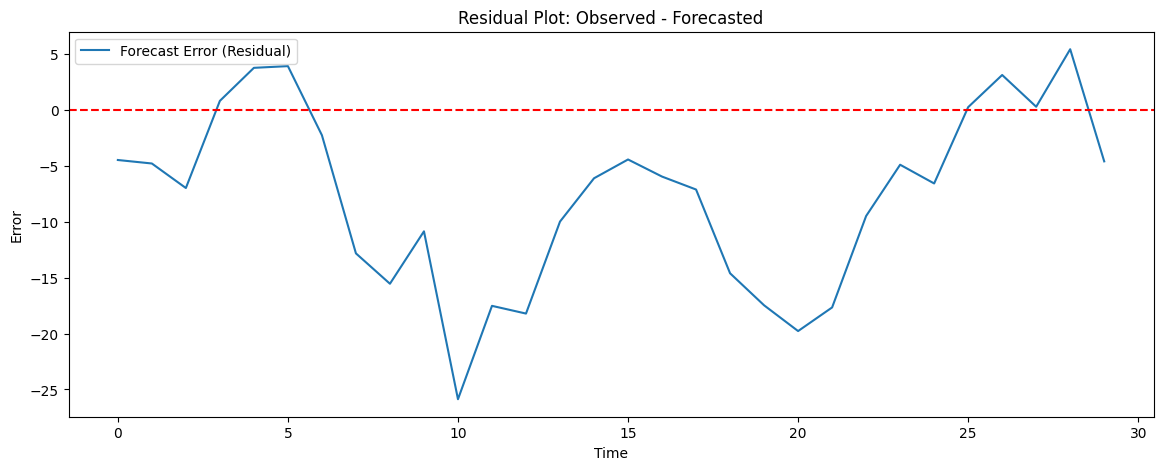
plt.ylabel('Error')

plt.legend()

plt.savefig("residual\_forecast\_plot.png")

plt.show()





6. Use grid search to optimize hyperparameters such as ndrop , ntimestep and batch size. Compare test result with previous findings.

from sklearn.model\_selection import ParameterGrid

hyperparameters = {

    "timesteps": [30, 60],

    "dropout\_rate": [0.2, 0.3],

    "batch": [16, 32]

}

optimal\_loss = float('inf')

optimal\_params = {}

for config in ParameterGrid(hyperparameters):

    x\_train, y\_train = create\_dataset(train\_data, config['timesteps'])

    x\_test, y\_test = create\_dataset(test\_data, config['timesteps'])

    x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

    x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1))

    model = Sequential()

    model.add(LSTM(50, return\_sequences=True, input\_shape=(config['timesteps'], 1)))

    model.add(Dropout(config['dropout\_rate']))

    model.add(LSTM(50, return\_sequences=True))

    model.add(Dropout(config['dropout\_rate']))

    model.add(LSTM(50))

    model.add(Dropout(config['dropout\_rate']))

    model.add(Dense(1))

    model.compile(optimizer='adam', loss='mean\_squared\_error')

    model.fit(x\_train, y\_train, epochs=5, batch\_size=config['batch'], verbose=0)

    predictions = model.predict(x\_test)

    predictions = scaler.inverse\_transform(predictions)

    y\_test\_actual = scaler.inverse\_transform(y\_test.reshape(-1, 1))

    mse = np.mean(np.square(predictions - y\_test\_actual))

    if mse < optimal\_loss:

        optimal\_loss = mse

        optimal\_params = config

print("Optimal Hyperparameters Discovered:")

print(f" - Timesteps: {optimal\_params['timesteps']}")

print(f" - Dropout Rate: {optimal\_params['dropout\_rate']}")

print(f" - Batch Size: {optimal\_params['batch']}")

print(f" - Test MSE: {optimal\_loss}")

Optimal Hyperparameters Discovered:

- Timesteps: 30

- Dropout Rate: 0.2

- Batch Size: 32

- Test MSE: 155.42212503305637