PREDIKSI HARGA SAHAM TLKM MENGGUNAKAN ALGORITMA LONG SHORT-TERM MEMORY

*Note: Sub-titles are not captured in Xplore and should not be used

1st Wafa Salma Sentanu
Universitas Multimedia Nusantara 00000057841

2nd Jeanet Wynne W. Kastilong
Universitas Multimedia Nusantara 00000057328

3rd Muhammad Alvin Versa R
Universitas Multimedia Nusantara 00000059444

***Abstract*—Dalam dunia investasi dan perdagangan pasar saham, pemahaman dalam menganalisis data merupakan hal yang sangat penting. Pasar saham memiliki pergerakan yang dinamis, sehingga diperlukan pemodelan data untuk melakukan prediksi harga saham dengan tingkat akurasi yang tinggi. Terdapat Beberapa algoritma yang dapat digunakan dalam memprediksi pasar saham salah satunya adalah algoritma Long-Short Term Memory. Penelitian ini fokus pada prediksi harga saham TLKM menggunakan algoritma Long Short-Term Memory (LSTM) dalam konteks investasi dan perdagangan pasar saham. LSTM, sebagai algoritma deep learning, dapat mengatasi prediksi pada data harga saham yang dinamis. Penelitian ini melibatkan pengumpulan data historis harga saham TLKM selama 20 tahun dan membaginya menjadi data training dan testing. Hasil eksperimen menunjukkan bahwa penggunaan LSTM memberikan prediksi harga saham yang akurat, memungkinkan investor dan pedagang saham untuk mengambil keputusan investasi yang lebih optimal. Penelitian ini memiliki dampak signifikan dalam pengambilan keputusan investasi di pasar saham.**

***Keywords:* analisis data*, deep learning, Long-Short Term Memory,* saham**

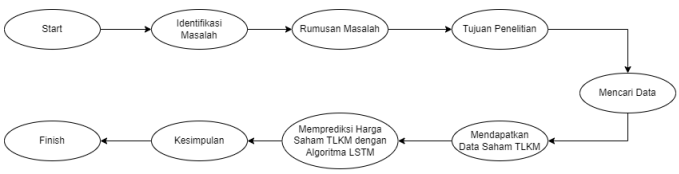
I. INTRODUCTION

Harga saham yang dimiliki oleh suatu perusahaan menunjukan nilai penyertaan dalam perusahaan. Pasalnya harga saham mencerminkan semua informasi yang tersedia secara umum di bursa maupun informasi lainnya yang hanya dapat diperoleh dari golongan - golongan tertentu untuk menjadikan pasar modal yang sempurna dan efisien. Pasar saham adalah pasar umum untuk perdagangan saham perusahaan & turunannya dengan harga yang disepakati. Memprediksi pasar saham bukanlah tugas yang mudah. Berbagai teknik digunakan dalam komoditas perdagangan untuk tugas prediksi[1]. Tinggi rendahnya harga saham dapat dipengaruhi oleh banyak faktor seperti kondisi dan kinerja perusahaan, resiko dividen, tingkat suku bunga, kondisi perekonomian, kebijaksanaan pemerintah, laju inflasi, penawaran dan permintaan, dsb[2]. Karena adanya kemungkinan perubahan dari faktor - faktor yang sudah disebutkan sebelumnya maka harga saham juga dapat mengalami fluktuasi harga setiap harinya.

Pasar saham dicirikan sebagai dinamis, tidak dapat diprediksi dan non-linear. Dengan demikian, untuk memaksimalkan keuntungan dan meminimalkan kerugian, teknik memprediksi nilai saham di terlebih dahulu dengan menganalisis tren selama beberapa tahun terakhir, hal ini terbukti sangat berguna untuk membuat pergerakan pasar saham [3].

PT. Telkom (TLKM) memiliki harga saham yang menjadi aset penting bagi para investor di pasar modal Indonesia. Kesempatan ini membuka peluang besar bagi perusahaan untuk menarik perhatian para investor atau pelaku pasar modal. Namun, dalam memilih keputusan investasi yang tepat maka diperlukannya prediksi fluktuasi harga saham TLKM[4]. Salah satu metode yang dapat digunakan untuk memprediksi fluktuasi harga saham yaitu menggunakan Algoritma LSTM (Long Short-Term Memory), salah satu algoritma deep learning ini mampu memproses data yang memiliki ketergantungan waktu atau disebut juga time series data[5]. Dengan menggunakan Algoritma LSTM, nantinya harga saham TLKM dapat menghasilkan model prediksi dengan akurasi yang lebih tinggi dibandingkan dengan metode-metode prediksi yang lainnya. Oleh karena itu, kami akan menerapkan Algoritma LSTM dalam memprediksi harga saham TLKM yang dimiliki oleh PT.Telkom guna membantu para investor dan pelaku pasar modal dalam mengambil keputusan investasi yang lebih tepat.

II. METHODOLOGY



Gambar 3. Metodologi Penelitian

Gambar diatas merupakan ilustrasi dari proses metode yang kami gunakan dalam penelitian ini. Pertama, penelitian ini akan melakukan sebuah identifikasi masalah dari topik yang sudah ditentukan yaitu memprediksi harga saham. Setelah proses identifikasi masalah selesai dilakukan,

selanjutnya merumuskan masalah tersebut dan menetapkan tujuan penelitian. Setelah itu, kami mencari data yang sesuai dengan topik yang kami bawakan yaitu mengenai prediksi harga saham. Akhirnya, kami mendapatkan data harga saham TLKM dari PT. Telkom Indonesia melalui https://finance.yahoo.com/. Dengan dataset ini kami akan melakukan sebuah prediksi menggunakan algoritma Long-Short Term Memory dan terdapat beberapa proses yang akan dijelaskan secara terperinci di Hasil dan Pembahasan. Lalu, kami membuat sebuah kesimpulan sebagai akhir dari penelitian ini.

Metode yang digunakan dalam penelitian ini melibatkan pengumpulan data historis harga saham TLKM selama kurang lebih 20 tahun dan membaginya menjadi data training dan testing. Data training digunakan untuk mempelajari pola dan tren dari data sebelum-sebelumnya, sementara data testing digunakan untuk menguji akurasi dalam prediksi model. Hasil eksperimen menunjukkan bahwa algoritma LSTM memberikan kinerja yang baik dalam menghasilkan prediksi harga saham TLKM yang akurat. Penelitian ini memiliki dampak yang signifikan dalam pengambilan keputusan pada investasi di pasar saham. Dengan menggunakan LSTM, investor dan pedagang saham dapat memperoleh prediksi harga saham yang akurat. Hal ini memungkinkan mereka untuk membuat keputusan dalam berinvestasi yang lebih optimal.

III. RESULTS AND DISCUSSION

A. Data Understanding

Memanggil dataset

```
In [3]: #Menginisialisasi dataset ke dalam variabel df
        df = pd.read_csv("TLK 2jan03-30des23.csv")

        #Untuk mengetahui values dari dataset
        df.isnull().values.any()
Out[3]: False
```

Menampilkan bagian atas dari dataset

```
In [4]: df.head()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2002-12-31	4.180	4.255	4.180	4.245	1.965326	61200
1	2003-01-02	4.225	4.225	4.160	4.175	1.932918	204800
2	2003-01-03	4.190	4.225	4.190	4.205	1.946807	159000
3	2003-01-06	4.000	4.145	3.985	4.105	1.900510	675800
4	2003-01-07	3.985	3.995	3.945	3.970	1.838008	635400

Gambar 4.1 Code import data

Setelah memanggil seluruh fungsi salah satunya library pandas, kami akan memanggil dataset harga saham TLKM dengan menggunakan fungsi read ke dalam variabel df dan menggunakan fungsi isnull().values untuk mengetahui values dari dataset seperti apa. Kemudian menggunakan head untuk menampilkan baris paling atas pada dataset yang dipanggil atau digunakan.

Menampilkan tipe data dari kolom dataset yang tersisa

```
In [10]: print(df.dtypes)

Date      object
Open     float64
High     float64
Low      float64
Close    float64
dtype: object

In [11]: df
```

Out[11]:

	Date	Open	High	Low	Close
0	2002-12-31	4.180000	4.255000	4.180000	4.245000
1	2003-01-02	4.225000	4.225000	4.160000	4.175000
2	2003-01-03	4.190000	4.225000	4.190000	4.205000
3	2003-01-06	4.000000	4.145000	3.985000	4.105000
4	2003-01-07	3.985000	3.995000	3.945000	3.970000
...	...	...	...	...	...
5031	2022-12-23	23.879999	24.040001	23.770000	24.030001
5032	2022-12-27	24.129999	24.139999	23.850000	23.900000
5033	2022-12-28	23.650000	23.770000	23.309999	23.420000
5034	2022-12-29	23.990000	23.990000	23.500000	23.680000
5035	2022-12-30	23.840000	23.889999	23.629999	23.850000

5036 rows × 5 columns

Gambar 4.2 Code tipe data kolom yang tidak dihapus

Menampilkan tipe data dari kolom yang tidak dihapus dan memanggil df untuk melihat data setelah memanggil tipe datanya.

B. Data Preparation

Melakukan scaling data (menggunakan kolom High)

```
In [629]: min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
          dataset = min_max_scaler.fit_transform(df['High'].values.reshape(-1, 1))
```

Gambar 4.3 Code minmax scaler

Gambar 4.3 digunakan untuk scaling data dengan menggunakan minmax scaler dengan feature rangenya antara 0 dan 1. Kemudian membuat variabel baru dataset untuk menyimpan fungsi fit transform (menghitung nilai minimum dan maksimum) dari data High.

Membagi data menjadi data train dan data test

```
In [630]: dataset[0:10]
Out[630]: array([[0.01798893],
                 [0.01706642],
                 [0.01706642],
                 [0.0146064 ],
                 [0.00999385],
                 [0.0101476 ],
                 [0.0146064 ],
                 [0.0149139 ],
                 [0.01476015],
                 [0.01752768]])

In [631]: train_size = int(len(dataset) * 0.8)
          test_size = len(dataset) - train_size
          train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
          print(len(train), len(test))
          4028 1008

In [632]: #Menampilkan panjang/jumlah dataframe
          print(len(df))
          5036
```

Gambar 4.4 Code pembagian data

Konversi value array kedalam dataset matriks

```
In [633]: def create_dataset(dataset, look_back=15):
              x, y = [], []
              for i in range(len(dataset)-look_back-1):
                  a = dataset[i:(i+look_back), 0]
                  x.append(a)
                  y.append(dataset[i + look_back, 0])
              return np.array(x), np.array(y)

In [634]: look_back = 15
          x_train, y_train = create_dataset(train, look_back)
          x_test, y_test = create_dataset(test, look_back)

In [635]: print(x_train.shape)
          print(y_train.shape)
          print(x_test.shape)
          print(y_test.shape)

          (4012, 15)
          (4012,)
          (992, 15)
          (992,)
```

Gambar 4.5 Code konversi value array ke matriks

Setelah scalling data maka akan dilakukan pembagian data train sebesar 80% dan test 20% (pada gambar 4.4). Kemudian pada gambar 4.5 membuat variable look_back berjumlah 15. Penggunaan LSTM membutuhkan input data (x) yang terstruktur ke dalam sample, time step, dan feature (gambar 4.6)

Reshape input menjadi 3D yaitu [samples, time step, dan feature]

```
In [636]: x_train = np.reshape(x_train, (x_train.shape[0], 1, x_train.shape[1]))
          x_test = np.reshape(x_test, (x_test.shape[0], 1, x_test.shape[1]))
```

Gambar 4.6 Reshape input menjadi 3 dimensi

C. *Modelling*

Membuat model LSTM

```
In [637]: look_back = 15

          model1 = Sequential()

          model1.add(LSTM(50, input_shape=(1, look_back), return_sequences=False))

          # Dropout to prevent overfitting
          model1.add(Dropout(0.5))

          # Dense layer for classification output
          model1.add(Dense(units=1, activation='tanh'))

          # Optimizing the neural network weights
          opt = tf.keras.optimizers.Adam(learning_rate=0.01)

          # Compiling the model with the optimizer
          model1.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

          history = model1.fit(x_train, y_train, epochs=100, batch_size=64, verbose=2, validation_data=(x_test, y

          Epoch 1/100
          63/63 - 3s - loss: 0.6912 - accuracy: 4.9850e-04 - val_loss: 0.6015 - val_accuracy: 0.0000e+00 -
          3s/epoch - 55ms/step
          Epoch 2/100
          63/63 - 0s - loss: 0.5996 - accuracy: 4.9850e-04 - val_loss: 0.5962 - val_accuracy: 0.0000e+00 -
          234ms/epoch - 4ms/step
          Epoch 3/100
          63/63 - 0s - loss: 0.5955 - accuracy: 4.9850e-04 - val_loss: 0.5937 - val_accuracy: 0.0000e+00 -
          241ms/epoch - 4ms/step
          Epoch 4/100
          63/63 - 0s - loss: 0.5946 - accuracy: 4.9850e-04 - val_loss: 0.5934 - val_accuracy: 0.0000e+00 -
          241ms/epoch - 4ms/step
          Epoch 5/100
          63/63 - 0s - loss: 0.5930 - accuracy: 4.9850e-04 - val_loss: 0.5931 - val_accuracy: 0.0000e+00 -
          244ms/epoch - 4ms/step
          Epoch 6/100
          63/63 - 0s - loss: 0.5921 - accuracy: 4.9850e-04 - val_loss: 0.5924 - val_accuracy: 0.0000e+00 -
          243ms/epoch - 4ms/step
          Epoch 7/100
```

Gambar 4.7 Code pemodelan LSTM

Setelah reshapping menjadi 3 dimensi , selanjutnya dibuat proses pemodelan (gambar 4.7) yang membutuhkan model layer by layer (layer lstm, dropout, dense). Untuk itu, kami menggunakan Sequential, optimizer:adam. Kemudian RMSE dan MSE dihasilkan seperti pada gambar 4.8.

```
In [645]: train_predict = model1.predict(x_train)
          test_predict = model1.predict(x_test)


          # #invert prediksi

          train_predict = min_max_scaler.fit_transform(train_predict)
          y_train = min_max_scaler.inverse_transform([y_train])

          test_predict = min_max_scaler.inverse_transform(test_predict)
          y_test = min_max_scaler.inverse_transform([y_test])

          #kalkulasi untuk root mean squared error

          train_score = math.sqrt(mean_squared_error(y_train[0], train_predict[:,0]))
          print('Train score: %.2f RMSE' % (train_score))

          test_score = math.sqrt(mean_squared_error(y_test[0], test_predict[:,0]))
          print('Test score: %.2f RMSE' % (test_score))

          #hitung MSE
          train_score1 = mean_squared_error(y_train[0], train_predict[:,0])
          print('Train score: %.2f MSE' % (train_score1))

          test_score1 = mean_squared_error(y_test[0], test_predict[:,0])
          print('Train score: %.2f MSE' % (test_score1))

          126/126 [==============================] - 1s 1ms/step
          31/31 [==============================] - 0s 2ms/step
          Train score: 0.07 RMSE
          Test score: 0.04 RMSE
          Train score: 0.01 MSE
          Train score: 0.00 MSE
```

Gambar 4.8 Code hasil RMSE dan MSE

D. *Evaluate*

```
In [640]: score = model1.evaluate(x_train, y_train, batch_size = 32, verbose = 2)
          print('Train Accuracy:', score[1])

          126/126 - 0s - loss: 0.0011 - rmse: 0.0328 - 168ms/epoch - 1ms/step
          Train Accuracy: 0.03278999775648117
```

Gambar 4.9 Code evaluasi RMSE dan MSE

Setelah membangun model lstm, model tersebut dievaluasi dengan mengggunakan model.evaluate.

Membuat tuning test dari hasil model yang telah dibuat dengan menggunakan perbandingan yang lainnya

```
In [641]: print("Shape x_train:", x_train.shape)
          # Output: (jumlah_sampel, timesteps, features)

          # Misalkan y_train adalah dataset target
          print("Shape y_train:", y_train.shape)
          # Output: (jumlah_sampel,)

          # Ambil nilai timesteps dari x_train
          timesteps = x_train.shape[1]
          print("Timesteps:", timesteps)

          # Ambil nilai features dari x_train
          features = x_train.shape[2]
          print("Features:", features)

          Shape x_train: (4012, 1, 15)
          Shape y_train: (4012,)
          Timesteps: 1
          Features: 15
```

Gambar 4.10 Code hasil shape, timesteps, dan features

Menginisialisasi shape, timesteps, dan features untuk keperluan dalam tuning test dalam melakukan perbandingan dari model yang sudah dibuat dengan model lainnya seperti pada codingan gambar 4.11.

```
In [644]: import numpy as np
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import LSTM, Dense
from keras.wrappers.scikit_learn import KerasClassifier

# Placeholder values for timesteps and features
timesteps = 1
features = 15

# Fungsi untuk membuat model LSTM
def create_model(units=50, activation='tanh', optimizer='opt'):
    model = Sequential()
    model.add(LSTM(units=units, activation=activation, input_shape=(timesteps, features)))
    model.add(Dense(1))
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

# Membuat objek KerasClassifier
model = KerasClassifier(build_fn=create_model)

# Mendefinisikan hyperparameter yang ingin diuji
param_grid = {
    'units': [50, 100, 200],
    'activation': ['tanh', 'sigmoid'],
    'optimizer': ['adam', 'rmsprop']
}

# Membuat objek GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5)

# Melakukan proses tuning dengan fit
grid_search.fit(x_train, y_train)

# Menampilkan parameter terbaik
print("Parameter terbaik:", grid_search.best_params_)

# Menampilkan skor akurasi terbaik
print("Skor akurasi terbaik:", grid_search.best_score_)

101/101 [==============================] - 3s 3ms/step - loss: -15694.9287 - accuracy: 6.2305e-04
26/26 [==============================] - 0s 2ms/step - loss: -13001.6670 - accuracy: 0.0000e+00
101/101 [==============================] - 2s 3ms/step - loss: -9649.2939 - accuracy: 9.3458e-04
26/26 [==============================] - 0s 2ms/step - loss: -20887.7910 - accuracy: 0.0000e+00
101/101 [==============================] - 2s 3ms/step - loss: 15494.6016 - accuracy: 3.1153e-04
26/26 [==============================] - 0s 2ms/step - loss: 29459.0215 - accuracy: 0.0000e+00
101/101 [==============================] - 3s 4ms/step - loss: -14028.3848 - accuracy: 3.1162e-04
26/26 [==============================] - 0s 2ms/step - loss: -270.1251 - accuracy: 0.0772
101/101 [==============================] - 3s 4ms/step - loss: 16351.4727 - accuracy: 3.1162e-04
26/26 [==============================] - 0s 2ms/step - loss: 15577.3359 - accuracy: 0.0000e+00
101/101 [==============================] - 3s 5ms/step - loss: -16033.8389 - accuracy: 6.2305e-04
26/26 [==============================] - 1s 3ms/step - loss: -13001.6670 - accuracy: 0.0000e+00
```

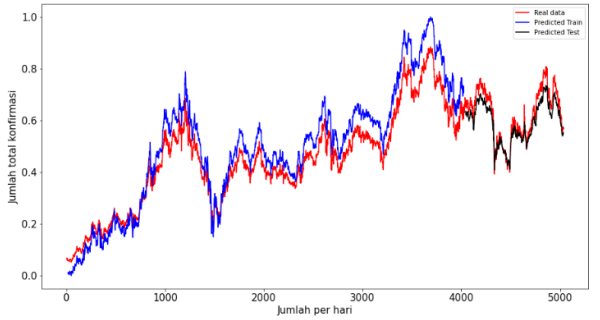
Gambar 4.11 Code hasil tuning test

E. Deployment

```
In [646]: #plotting untuk train prediction
train_predict_plot = np.empty_like(dataset)
train_predict_plot[:, :] = np.nan
train_predict_plot[look_back:len(train_predict)+look_back, :] = train_predict

# plotting untuk test prediction
test_predict_plot = np.empty_like(dataset)
test_predict_plot[:, :] = np.nan
test_predict_plot[len(train_predict)+(look_back*2)+1:len(dataset)-1, :] = test_predict

#plot
plt.figure(figsize=(15, 8))
plt.plot(min_max_scaler.inverse_transform(dataset), color='r', label='Real data')
plt.plot(train_predict_plot, color='b', label='Predicted Train')
plt.plot(test_predict_plot, color='black', label = 'Predicted Test')
plt.xticks(size=15)
plt.yticks(size=15)
plt.xlabel("Jumlah per hari", size=15)
plt.ylabel("Jumlah total konfirmasi", size=15)
plt.legend(loc='best')
plt.show()
```



Gambar 4.12 Code visualisasi model LSTM

Pada gambar 4.12 merupakan hasil visualisasi dari perbandingan data asli dan prediksi menggunakan lstm.

F. Diskusi

Dari hasil pembahasan di atas berdasarkan pembuatan, evaluasi, dan implementasi model yang dilakukan, dapat dikatakan penggunaan LSTM baik digunakan dalam melakukan prediksi saham. Karena selain mempunyai banyak parameter, terdapat banyak activate dan optimizer yang dapat digunakan untuk mendapatkan model yang lebih baik. Dalam hal saham TLKM, activation dan optimizer yang digunakan yaitu ‘tanh’ dan ‘adam’. Dan menghasilkan akurasi mse dan rsme adalah 0,0327 dst serta train dan test score mse & rmse berjumlah (train rmse: 0.07, test rmse: 0.04, train mse: 0.01, test mse: 0.00) walaupun score-score tersebut belum mencapai hasil yang seharusnya namun dapat dikatakan cukup baik untuk diterapkan karena didukung juga oleh output penerapan tuning test.

IV. CONCLUSION

Uji coba yang telah dilakukan menggunakan dataset TLKM memiliki 5036 baris (2 Januari 2003 - 30 Desember 2022) menunjukkan hasil yang cukup baik dengan error dan loss yang tergolong kecil. Tampilan tersebut dihasilkan dengan melihat data yang sudah lalu sebanyak 15 hari dengan menggunakan perbandingan train dan test nya 50:50, epochs: 100 dengan fungsi aktivasi Tunh. Dari hasil pembahasan di atas berdasarkan pembuatan, evaluasi, dan implementasi model yang dilakukan, dapat dikatakan penggunaan LSTM baik digunakan dalam melakukan prediksi saham. Karena selain mempunyai banyak parameter, terdapat banyak activate dan optimizer yang dapat digunakan untuk mendapatkan model yang lebih baik. Dalam hal saham TLKM, activation dan optimizer yang digunakan yaitu ‘tanh’ dan ‘adam’. Dan menghasilkan akurasi mse dan rsme adalah 0,0327 dst serta train dan test score mse & rmse berjumlah (train rmse: 0.07, test rmse: 0.04, train mse: 0.01, test mse: 0.00) walaupun score-score tersebut belum mencapai hasil yang seharusnya namun dapat dikatakan cukup baik untuk diterapkan karena didukung juga oleh output penerapan tuning test.

References

[1] R.K. Dase, D. D. Pawar. (2010). “Application of Artificial Neural Network for stock market predictions: A review of literature”. International Journal of Machine Intelligence, ISSN: 0975–2927, Volume 2, Issue 2, 2010, pp-14-17

[2] V. Mehar, C. Deeksha, A. T. Vinay, K. Arun. (2020). “Stock Closing Price Prediction using Machine Learning Techniques”. Procedia Computer Science 167 (2020) 599–606.

[3] F. M. Islam and A. D. Prasetyo, “Financial Feasibility Study for new investment in new digital product of PT Telkom Indonesia (case study: SKP project),” *European Journal of Business and Management Research*, vol. 5, no. 5, Oct. 2020.

[4] M. L. Kitri and C. E. Manurung, “Impact of stock repurchase announcement on Indonesia Stock Market Reaction 2015-2019,” *Jurnal Ilmu Sosial Politik dan Humaniora*, vol. 3, no. 2, pp. 32–40, 2020.

[5] H. T. Pham, T. Q. Nguyen, and T. M. Nguyen, "LSTM-Based Time Series Anomaly Detection with

[6] Application to Retail Data," IEEE Trans. Ind. Informatics, vol. 15, no. 6, pp. 3424–3432, Jun. 2019.

[7] M. A. Al-Qassas, N. M. Al-Ma’adeed, and M. S. Al-Razgan, "Predicting Blood Glucose Levels in Diabetic Patients using LSTM Neural Networks," IEEE J. Biomed. Health Informatics, vol. 23, no. 3, pp. 1273–1283, May 2019.

[8] A. M. Khalil, R. P. Singh, and G. S. Sodhi, "Predicting Water Quality Parameters using LSTM Neural Networks," Environ. Sci. Pollut. Res., vol. 27, no. 31, pp. 38684–38694, Nov. 2020.

[9] C. Yuan, Y. Chen, X. Cao, Z. Wu, and L. Chen, "Exploring the Capabilities of LSTM Neural Networks for the Prediction of Runoff and Sediment Yield," J. Hydrol., vol. 602, p. 126448, Mar. 2021.

[10] W. Xu, J. Yang, H. Yu, Y. Li, and B. Liu, "LSTM-based Credit Card Fraud Detection using Synthetic Minority Over-sampling Technique," IEEE Access, vol. 9, pp. 43279–43287, Mar. 2021.