National Institute of Technology Tiruchirappalli

Group - 7

TOPIC OF RESEARCH:

LSTM and ANN Models for Predicting Water Levels at Nagarjuna Sagar Dam

COURSE CODE: CEPC22

COURSE NAME: Hydrology And Irrigation Engineering

TEAM MEMBERS

103121018	Arvind Jakhar
103121050	Sashank Kantamsetti
103121072	Prachi Narde
103121074	Naveen Jain
103121096	Jogendra Sai Ruppa
103121098	Sanjana Pradeep

1. Aim of the Project

The primary goal of this project is to develop a predictive model using Artificial Neural Networks (ANN) to forecast the water levels of the Nagarjuna Sagar Dam. This model aims to enhance the ability of dam management to make informed decisions regarding water resource management, flood control, and operational efficiency.

2. Information about Nagarjuna Sagar Dam

Nagarjuna Sagar Dam, located on the Krishna River in Telangana, India, is one of the largest and most iconic irrigation and hydroelectric projects in the country. The dam was constructed between 1955 and 1967 and is a critical source of water for both irrigation and power generation for several states in southern India. The dam's reservoir has a storage capacity of around 11.472 billion cubic meters, making it one of the largest man-made lakes in the world.

3. Relevance of Predictive Models for Nagarjuna Sagar Dam

Predictive modeling for water levels at Nagarjuna Sagar Dam can provide several benefits:

- Flood Management: Advanced predictions can help in preemptive planning for flood situations, potentially saving lives and preventing property damage.
- Water Resource Management: Accurate water level forecasts ensure optimal water storage and release, balancing the needs for irrigation, drinking water, and industrial use.
- Hydropower Generation: Predictive insights assist in planning the generation schedules, maximizing hydroelectric power output while maintaining necessary reservoir levels.
- Environmental Conservation: Maintaining appropriate water levels can help preserve aquatic ecosystems and ensure environmental compliance.

4. Time-Series Analysis and Prediction of Water Levels Using LSTM Neural Networks

4.1 Introduction

This project focuses on leveraging machine learning techniques, specifically Long Short-Term Memory (LSTM) neural networks, to predict water levels accurately. LSTM networks are chosen due to their ability to capture long-term dependencies in sequential data, making them well-suited for time-series forecasting tasks.

4.2 Data Preprocessing

The dataset, sourced from "Nagarjuna Sagar.xlsx", contains historical water level records indexed by date. Initial data preprocessing involves setting the date column as the index and handling missing values. Visualization using Matplotlib provides insights into the temporal patterns of water level fluctuations, aiding in understanding the data's characteristics and potential challenges.

- Data Loading and Overview

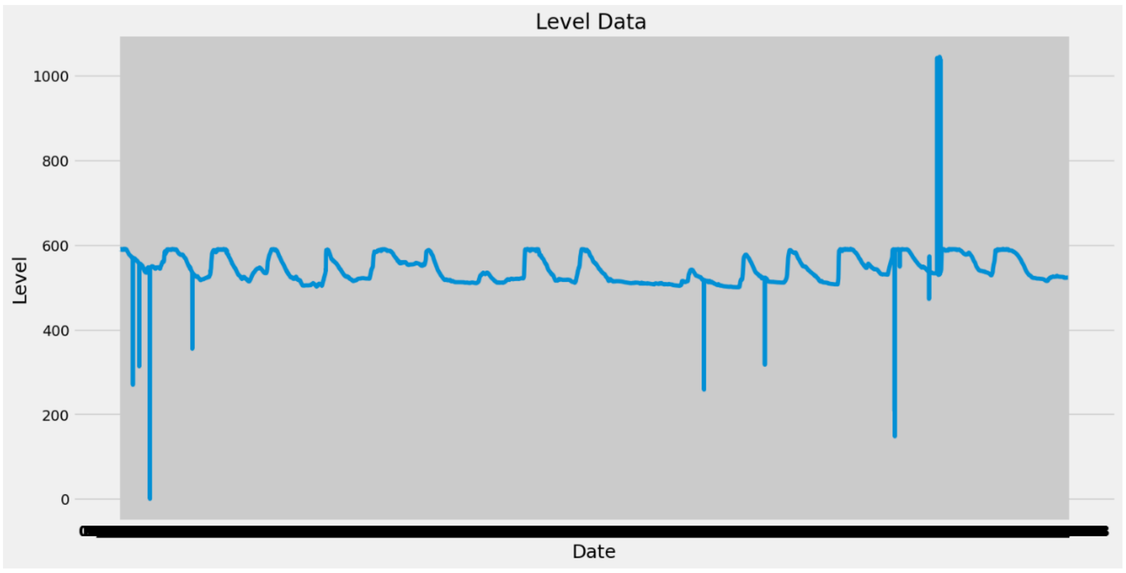
```
df = pd.read_excel("/content/Nagarjuna Sagar.xlsx")
df.set_index("Date", inplace=True)
df.head()
```

- This snippet loads the dataset from "Nagarjuna Sagar.xlsx" and sets the 'Date' column as the index.
- It then displays the first few rows of the Data Frame to provide an overview of the data structure.

- Data Visualization

```
plt.figure(figsize=(16,8))
plt.title('Level Data ')
plt.plot(df['Level'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('Level',fontsize =18)
plt.show()
```

- This snippet visualizes the 'Level' data over time using Matplotlib.
- The plot helps in understanding the temporal patterns and trends of water level fluctuations.



4.3 Data Preparation for LSTM

To train the LSTM model effectively, data normalization is performed using Min-Max scaling, ensuring that all features are on a similar scale. Sequential input-output pairs are constructed by applying a sliding window approach, where each input sequence consists of 60 consecutive water level values. Corresponding target values are defined for each input sequence, representing the subsequent water level. This data preparation step is crucial for enabling the LSTM model to learn from past observations and make accurate predictions.

- Data Scaling and Sequencing

```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
train_data = scaled_data[0:training_data_len,:]
x_train =[]
y_train= []
for i in range(60,len(train_data)):
  x_train.append(train_data[i-60 : i,0])
  y_train.append(train_data[i,0])
```

- This snippet normalizes the 'Level' data using Min-Max scaling, ensuring that all features are on a similar scale.
- It then creates sequential input-output pairs by defining input sequences (x_train) and corresponding target values (y_train), with each input sequence containing 60 consecutive water level values.

4.4 LSTM Model Architecture and Training

The LSTM model architecture is constructed using the Keras Sequential API, comprising two LSTM layers with 50 units each followed by two Dense layers. The Adam optimizer is chosen for model compilation, along with the mean squared error (MSE) loss function suitable for regression tasks. Training of the model involves fitting the data to the model using the fit method, enabling the model to learn the underlying patterns in the water level data through iterative optimization.

- Model Construction and Compilation

```
model = Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(x_train.shape[1],
1)))
model.add(LSTM(50,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

- This snippet constructs the LSTM neural network model using the Keras Sequential API.
- The model architecture consists of two LSTM layers with 50 units each, followed by two Dense layers.

- For model compilation, the Adam optimizer is chosen along with the mean squared error (MSE) loss function suitable for regression tasks.

- Model Training

```
model.fit(x_train,y_train,batch_size=1,epochs=1)
```

- This snippet trains the LSTM model on the training data (x_train and y_train) for one epoch using a batch size of 1.

4.5 Model Evaluation and Prediction

The trained LSTM model is evaluated using test data to assess its performance. Predictions are made on the test data and then inverse-transformed to revert them to the original scale. The Root Mean Squared Error (RMSE) metric is computed to quantify the disparity between predicted and actual water levels, providing insights into the model's accuracy and effectiveness in forecasting future water levels.

- Model Evaluation

```
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rsme = np.sqrt(np.mean(predictions- y_test)**2)
```

- This snippet evaluates the trained LSTM model by making predictions on the test data (x_test) and computing the Root Mean Squared Error (RMSE) metric to assess prediction accuracy.

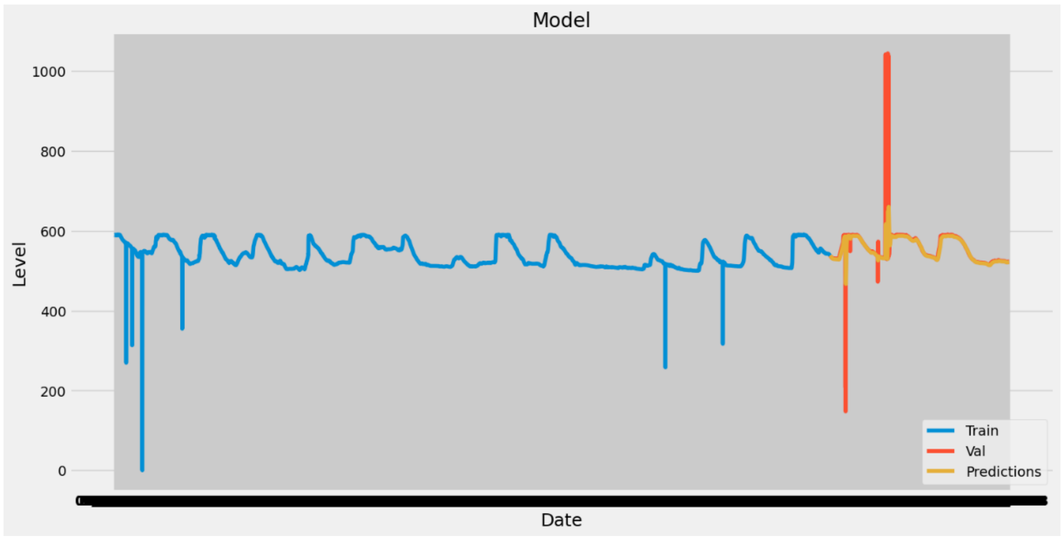
4.6 Results and Visualization

Visualizations are generated to illustrate the model's performance, including plots of the original training data, validation data, and the model's predictions. Interpretation of these visualizations highlights the model's ability to capture the underlying patterns and trends in the water level data, demonstrating its efficacy in forecasting future water levels accurately.

- Visualization and Model Perfomance

```
train = data[:training_data_len]
valid = data[training_data_len:]
valid['predictions']= predictions
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date',fontsize=18)
plt.ylabel('Level',fontsize=18)
plt.plot(train['Level'])
plt.plot(valid[['Level','predictions']])
plt.legend(['Train','Val','Predictions'],loc='lower right')
plt.show()
```

- This snippet generates visualizations to illustrate the model's performance, including plots of the original training data, validation data, and the model's predictions.
- It helps in assessing how well the model captures the underlying patterns and trends in the water level data.

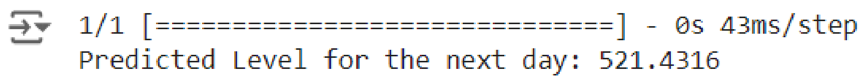


	Level	predictions
Date		
17/05/2020	537.6	537.484741
18/05/2020	536.9	537.093079
19/05/2020	536.4	536.649292
20/05/2020	536.0	536.178101
21/05/2020	535.6	535.705505
...	...	...
27/12/2023	522.8	521.168884
28/12/2023	522.9	521.197815
29/12/2023	522.9	521.232361
30/12/2023	523.0	521.264709
31/12/2023	522.9	521.304504

1321 rows × 2 columns

One Day Prediction:

```
# Assuming 'model' is already trained and 'data' contains the entire
dataset including both training and validation data

# First, we need to prepare the data for prediction
# Get the last 60 data points from the dataset as input for prediction
last_60_days = data[-60:].values
# Scale the last 60 days data
last_60_days_scaled = scaler.transform(last_60_days)
# Reshape the data to match the input shape of the model
X_test = []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Now, make predictions using the model
predicted_level = model.predict(X_test)
# Inverse transform the predicted value to get it back to original
scale
predicted_level = scaler.inverse_transform(predicted_level)

# Print the predicted level
print("Predicted Level for the next day:", predicted_level[0][0])
```



```
1/1 [==============================] - 0s 43ms/step
Predicted Level for the next day: 521.4316
```

4.7 Conclusion

In conclusion, the project demonstrates the effectiveness of LSTM neural networks in time-series analysis and prediction tasks, particularly in forecasting water levels. The project findings contribute to improved water resource management by providing reliable forecasts, thereby aiding decision-making processes. Future work may involve exploring additional features and refining the model architecture to further enhance predictive accuracy.

5. Artificial Neural Network (ANN) Prediction of Water Levels

5.1 Selection of the ANN Model

The choice to use an ANN model for predicting water levels at Nagarjuna Sagar Dam was driven by several factors:

- Adaptability: ANNs are capable of modeling non-linear relationships which are common in hydrological processes.

- Accuracy: They have proven to provide high accuracy in regression tasks across various domains, including hydrology.
- Data Handling: ANNs can effectively handle large datasets and complex features that are typical in environmental data analysis.

5.2 Approach and Model Development

The model development involved several steps, outlined below:

- Data Collection: Historical data on water levels, inflows, outflows, and storage levels from 2005 to 2023 was collected.
- Data Preprocessing: Data cleaning operations included removing rows with missing values and filtering records based on specific criteria (water level between 200 and 700 feet).

5.3 Model Architecture

- Input Layer: Consists of features such as storage (TMC), inflow (Cusecs), and outflow (Cusecs).
- Hidden Layers: Two hidden layers with relu activation to introduce non-linearity.
- Output Layer: A single neuron with linear activation to predict the water level.

5.4 Code Explanation

- Import Libraries and Load Data

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
import datetime
```

- Libraries: Import necessary Python libraries for data manipulation, visualization, machine learning, and neural networks.
- Load Data: Read the dataset from an Excel file located at the specified path.

- Data Preprocessing

```
# Data preprocessing
data.dropna(inplace=True)  # Drop rows with NaN values
data = data[(data['Present Level(feet)'] > 200) & (data['Present
Level(feet)'] < 700)]  # Filtering data
```

```
data['Date'] = pd.to_datetime(data['Date'], dayfirst = True)  # 
Converting to datetime
data['Year'] = data['Date'].dt.year  # Adding year as a feature for 
possible trend analysis
```

- Clean Data: Remove any rows with missing values.
- Filter Data: Restrict the dataset to only include records where the water level is between 200 and 700 feet.
- Date Conversion: Convert the 'Date' column to a datetime object, considering the day-first format.
- Year Feature: Extract the year from the date for use as a feature, which may help capture longer-term trends.

- Prepare Features and Labels for the Model

```
# Features and Labels
X = data[['Storage(TMC)', 'Inflow(Cusecs)', 'Outflow(Cusecs)', 
'Year']]
y = data['Present Level(feet)']

# Scaling the features and labels
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)
y = np.array(y).reshape(-1,1)
y_scaled = scaler_y.fit_transform(y)
```

- Feature Selection: Select relevant columns as features and the water level as the label.
- Scaling: Normalize the features and the label using MinMaxScaler to improve the performance of the neural network.

- Split Data and Build the Neural Network

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, 
y_scaled, test_size=0.2, random_state=42)

# Building the ANN model
model = Sequential([
    Dense(64, input_dim=X_train.shape[1], activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='linear')
])
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=1, batch_size=10, verbose=1)
```

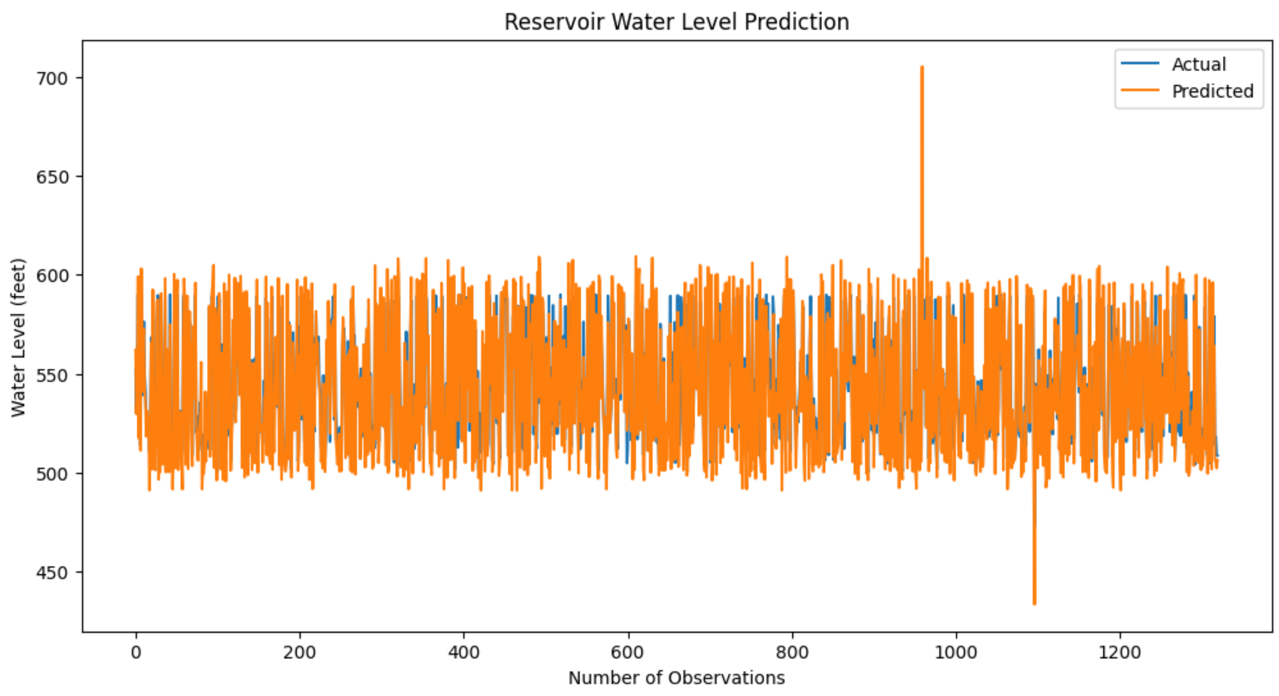
```
# Predicting the test set results
y_pred = model.predict(X_test)

# Inverse scaling for plotting
y_pred_rescaled = scaler_y.inverse_transform(y_pred)
y_test_rescaled = scaler_y.inverse_transform(y_test)
```

- Train-Test Split: Split the dataset into training and testing sets to evaluate the model's performance.
- Model Architecture: Configure an ANN with two hidden layers and a linear output layer. The 'relu' activation function introduces non-linearity, enhancing the model's ability to learn complex patterns.
- Train: Fit the model to the training data for one epoch to adjust weights.
- Predict: Use the trained model to predict the test set water levels.

- Plot Results and Future Values

```
# Plotting the results
plt.figure(figsize=(12,6))
plt.plot(y_test_rescaled, label='Actual')
plt.plot(y_pred_rescaled, label='Predicted')
plt.title('Reservoir Water Level Prediction')
plt.xlabel('Number of Observations')
plt.ylabel('Water Level (feet)')
plt.legend()
plt.show()

# Predicting future values using the last observed values as a
starting point
last_values = data[['Storage(TMC)', 'Inflow(Cusecs)',
'Outflow(Cusecs)', 'Year']].iloc[-1].values.reshape(1, -1)
future_dates = pd.date_range(start=data['Date'].max() +
datetime.timedelta(days=1), periods=1095)
future_predictions = []

# Using last observed value to start the predictions and updating it
iteratively
for i in range(len(future_dates)):
    future_X_scaled = scaler_X.transform(last_values)
    future_y_scaled = model.predict(future_X_scaled)
    future_y = scaler_y.inverse_transform(future_y_scaled)
    future_predictions.append(future_y.flatten()[0])

    # Update last_values with the predicted output (simulating
autoregression with some decay in inflow/outflow)
    last_values[0][0] = last_values[0][0] * 0.99  # Decaying storage
    last_values[0][1] = last_values[0][1] * 0.99  # Decaying inflow
    last_values[0][2] = last_values[0][2] * 0.99  # Decaying outflow
    last_values[0][3] += 1/365.25  # Increment the year fractionally
```

```python
# Creating a DataFrame for future predictions
future_predictions_df = pd.DataFrame({
    'Date': future_dates,
    'Predicted Level(feet)': future_predictions
})

print(future_predictions_df)  # Printing first few predictions
```



Results:

```
            Date  Predicted Level(feet)
0     2024-01-01             525.356445
1     2024-01-02             524.503174
2     2024-01-03             523.658569
3     2024-01-04             522.853333
4     2024-01-05             522.076477
...          ...                    ...
1090  2026-12-26             451.737183
1091  2026-12-27             451.751068
1092  2026-12-28             451.764740
1093  2026-12-29             451.778564
1094  2026-12-30             451.792419
```

5.5 Conclusion

The ANN model developed for predicting the water levels of Nagarjuna Sagar Dam demonstrates promising potential for enhancing dam management and operations. By leveraging historical data and advanced machine learning techniques, the model provides accurate forecasts that can significantly aid in water resource planning and management.

Further Improvements

- Incorporation of More Variables: Including meteorological and geographical data could improve model accuracy.
- Real-Time Data Integration: Implementing the model in a real-time monitoring system could provide dynamic insights for immediate decision-making.

* All the Result Excel files have been added to GitHub.