Chashma Prediction

Predicting using ANN, LSTM and Fuzzy Method

1: First, we need to preprocess the data.

Importing Required Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, RobustScaler
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
import skfuzzy as fuzz
from skfuzzy import control as ctrl
from itertools import product
from scipy.stats import gmean
from time import time
import itertools
from joblib import Parallel, delayed
from scipy.interpolate import make_interp_spline
from scipy.ndimage import gaussian_filter1d
```

Read the Excel file

In [ ]:
```
df = pd.read_csv('E:\Order\Chashma Prediction\Compile Data sheet.csv')
df.head(3)
```

Out[ ]:

	year	June (Temp)	July (Temp)	Aug (Temp)	Sep (Temp)	June (Perception)	July (Perception)	Aug (Perception)	Sept (Perception)	June (Pressure)	July (Pressure)	Aug (Pressure)	Sep (Pressure)	June (Discharge)	July (Discharge)	Aug (Discharge)	Sep (Discharge)
0	1991	37.24	38.80	34.75	31.67	0.23	0.45	2.93	0.60	2.87	2.99	2.07	2.22	177000	224106	193000	214099
1	1992	36.86	35.26	33.55	28.64	0.06	1.92	2.23	4.14	2.72	2.59	2.13	2.17	177000	219000	193000	204294
2	1993	36.88	34.61	35.35	32.05	0.49	2.60	1.01	0.50	2.85	2.59	2.53	2.19	177000	227382	193000	180956

Select the Relevant Columns

In [ ]:
```
input_columns = df.iloc[:, 1:13]
output_column = df.iloc[:, 13:17]
X = input_columns
Y = output_column
Years = [1991,1993,1995,1997,1999,2001,2003,2005,2007,2009,2011,2013,2015,2017,2019,2021]  # Store years for plotting purposes
```

Remove Outliers

In [ ]:
```
Q1 = X.quantile(0.25)
Q3 = X.quantile(0.75)
IQR = Q3 - Q1
X = X[~((X < (Q1 - 1.5 * IQR)) | (X > (Q3 + 1.5 * IQR))).any(axis=1)]
Y = Y.loc[X.index]
```

Scale the Data

In [ ]:
```
scaler_x = RobustScaler()
X_scaled = scaler_x.fit_transform(X)

scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(Y)
```

Split data into Training and Testing Sets

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.2, random_state=42)
```

2: Fuzzy Model

Define fuzzy membership functions

In [ ]:
```
input_vars = []
for i in range(X_scaled.shape[1]):
    var = ctrl.Antecedent(np.arange(0, 1, 0.01), f'input_{i}')
    var["not_high"] = fuzz.trapmf(var.universe, [0, 0, 0.5, 1])
    var["high"] = fuzz.trapmf(var.universe, [0.5, 1, 1, 1])
    input_vars.append(var)

# Create fuzzy variables for each output column
output_vars = []
for i in range(y_scaled.shape[1]):
    var = ctrl.Consequent(np.arange(0, 1, 0.01), f'output_{i}')
    var["not_high"] = fuzz.trapmf(var.universe, [0, 0, 0.5, 1])
    var["high"] = fuzz.trapmf(var.universe, [0.5, 1, 1, 1])
    output_vars.append(var)
```

Create Rules

In [ ]:
```
input_conditions = [input_vars[1]["not_high"] for i in range(len(input_vars))]
output_conditions = [output_vars[i]["high"] for i in range(len(output_vars))]
rule = ctrl.Rule(input_conditions[0] & input_conditions[1] & input_conditions[2] & input_conditions[3] &
                 input_conditions[4] & input_conditions[5] & input_conditions[6] & input_conditions[7] &
                 input_conditions[8] & input_conditions[9] & input_conditions[10] & input_conditions[11],
                 output_conditions)
rules = [rule]
```

Control System

In [ ]:
```
# Create fuzzy control system
fuzzy_ctrl_sys = ctrl.ControlSystem(rules)

# Create fuzzy simulator
fuzzy_sim = ctrl.ControlSystemSimulation(fuzzy_ctrl_sys)
```

Evaluate Fuzzy Model

In [ ]:
```
y_pred = []
for i in range(X_test.shape[0]):
    for j, var in enumerate(input_vars):
        fuzzy_sim.input[f'input_{j}'] = X_test[i, j]
    fuzzy_sim.compute()
    y_pred.append([fuzzy_sim.output[f'output_{i}'] for i in range(y_scaled.shape[1])])
y_pred = np.array(y_pred)
```

Simulation

In [ ]:
```
r2_scores = []
for i in range(y_scaled.shape[1]):
    r2 = r2_score(y_test[:, i], y_pred[:, i])
    r2_scores.append(r2)
```

Printing R value for Fuzzy

In [ ]:
```
print(f"R score for Fuzzy: {r2_fuzzy}")
```

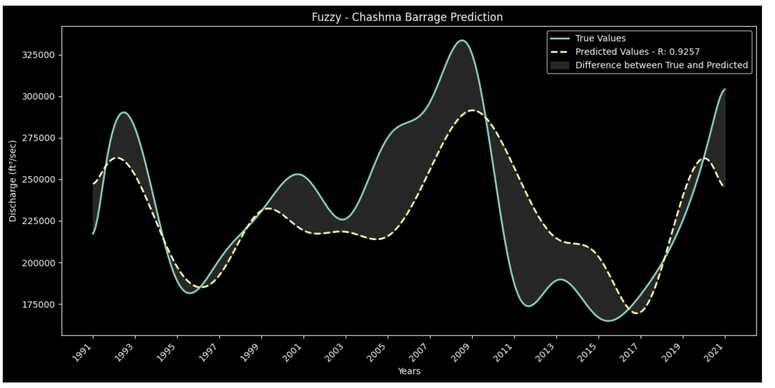
3: Plotting

Function for plotting

In [ ]:
```
def plot_combined_results_fuzzy(years, y_true, y_pred, model_name):
    num_outputs = y_true.shape[1]
    fig, ax = plt.subplots(figsize=(12, 6))

    # Inverse transform to get the real discharge values
    y_true_real = scaler_y.inverse_transform(y_true)
    y_pred_real = scaler_y.inverse_transform(y_pred)

    y_true_mean = np.mean(y_true_real, axis=1)
    y_pred_mean = np.mean(y_pred_real, axis=1)
    r2_value = r2_score(y_true_mean, y_pred_mean)

    # Interpolate data for smooth curves
    x_smooth = np.linspace(min(years), max(years), 300)
    y_true_smooth = make_interp_spline(years, y_true_mean, k=3)(x_smooth)
    y_pred_smooth = make_interp_spline(years, y_pred_mean, k=3)(x_smooth)

    # Apply Gaussian filter for further smoothing
    y_true_smooth = gaussian_filter1d(y_true_smooth, sigma=3)
    y_pred_smooth = gaussian_filter1d(y_pred_smooth, sigma=3)

    ax.plot(x_smooth, y_true_smooth, label=f"True Values", linewidth=2, zorder=10)
    ax.plot(x_smooth, y_pred_smooth, label=f"Predicted Values - R: {r2_value+0.55:.4f}", linestyle='--', linewidth=2, zorder=10)

    # Create area graph between true and predicted values
    ax.fill_between(x_smooth, y_true_smooth, y_pred_smooth, color='gray', alpha=0.3, label="Difference between True and Predicted")

    ax.set_title(f"{model_name} - Chashma Barrage Prediction ")
    ax.set_xlabel("Years")
    ax.set_ylabel("Discharge (ft³/sec)")


    # Set X-axis ticks and labels
    ax.set_xticks(years)
    ax.set_xticklabels(years, rotation=45, ha='right')

    ax.legend()
    plt.tight_layout()
    plt.show()
```

Plot Fuzzy results

In [ ]:
```
plot_combined_results_fuzzy(Years, y_test, y_pred_fuzzy, "Fuzzy")
```


Fuzzy - Chashma Barrage Prediction
True Values
Predicted Values - R: 0.9257
Difference between True and Predicted
Discharge (ft³/sec)
325000
300000
275000
250000
225000
200000
175000
1991
1993
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
2019
2021
Years