# Import Libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM, SimpleRNN

import matplotlib.pyplot as plt

# Load and Preprocess the Data

data = pd.read\_csv('electricity\_bill\_data.csv') # Replace with your dataset

# Assuming the dataset has features like 'Month', 'Temperature', 'Units\_Consumed', etc.

X = data[['Month', 'Temperature', 'Units\_Consumed']].values

y = data['Bill\_Amount'].values

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature scaling

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Support Vector Machine (SVM)

svm\_model = SVR(kernel='rbf')

svm\_model.fit(X\_train, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test)

svm\_mse = mean\_squared\_error(y\_test, y\_pred\_svm)

print(f'SVM Mean Squared Error: {svm\_mse}')

# Artificial Neural Network (ANN)

ann\_model = Sequential()

ann\_model.add(Dense(units=32, activation='relu', input\_shape=(X\_train.shape[1],)))

ann\_model.add(Dense(units=64, activation='relu'))

ann\_model.add(Dense(units=1))

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

ann\_model.fit(X\_train, y\_train, epochs=50, batch\_size=10, verbose=1)

y\_pred\_ann = ann\_model.predict(X\_test)

ann\_mse = mean\_squared\_error(y\_test, y\_pred\_ann)

print(f'ANN Mean Squared Error: {ann\_mse}')

# Recurrent Neural Network (RNN)

X\_train\_rnn = X\_train.reshape((X\_train.shape[0], 1, X\_train.shape[1]))

X\_test\_rnn = X\_test.reshape((X\_test.shape[0], 1, X\_test.shape[1]))

rnn\_model = Sequential()

rnn\_model.add(SimpleRNN(units=50, activation='relu', input\_shape=(X\_train\_rnn.shape[1], X\_train\_rnn.shape[2])))

rnn\_model.add(Dense(units=1))

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

rnn\_model.fit(X\_train\_rnn, y\_train, epochs=50, batch\_size=10, verbose=1)

y\_pred\_rnn = rnn\_model.predict(X\_test\_rnn)

rnn\_mse = mean\_squared\_error(y\_test, y\_pred\_rnn)

print(f'RNN Mean Squared Error: {rnn\_mse}')

# Long Short-Term Memory (LSTM)

lstm\_model = Sequential()

lstm\_model.add(LSTM(units=50, activation='relu', input\_shape=(X\_train\_rnn.shape[1], X\_train\_rnn.shape[2])))

lstm\_model.add(Dense(units=1))

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

lstm\_model.fit(X\_train\_rnn, y\_train, epochs=50, batch\_size=10, verbose=1)

y\_pred\_lstm = lstm\_model.predict(X\_test\_rnn)

lstm\_mse = mean\_squared\_error(y\_test, y\_pred\_lstm)

print(f'LSTM Mean Squared Error: {lstm\_mse}')

# Compare Results

models = ['SVM', 'ANN', 'RNN', 'LSTM']

mse\_values = [svm\_mse, ann\_mse, rnn\_mse, lstm\_mse]

plt.bar(models, mse\_values, color=['blue', 'green', 'orange', 'red'])

plt.ylabel('Mean Squared Error')

plt.title('Comparison of Models')

plt.show()