Project Report

For

AdaptiPlan: Intelligent Scenario Modelling for Climate Change Mitigation using Computational Statistical Model

5th November, 2024

Prepared by

Specialization	**SAP ID**	**Name**
AIML HONS.	500090912	CHARU GUPTA
AIML HONS.	500094127	LAKSHAY AGARWAL

Artificial Intelligence Cluster

School of Computer Science

UPES Dehradun – 248007, Uttarakhand

Table of Contents

1. ABSTRACT:

This project aims to develop a comprehensive web application that combines Al-driven scenario planning with a comparative analysis of climate change adaptation strategies. By integrating time series forecasting (ARIMA) and Monte Carlo simulations, the application will simulate various climate change scenarios and evaluate the effectiveness of different adaptation measures.

2. INTRODUCTION:

Climate change poses significant risks to ecosystems, economies, and societies globally. Accurate forecasting of climate variables such as temperature, precipitation, and extreme weather events is crucial for developing effective mitigation strategies. Traditional climate models, while robust, often require substantial computational resources and may not capture localized patterns effectively.

The primary purpose of the "AdaptiPlan" project is to develop a predictive tool that helps model future climate change scenarios using a combination of advanced machine learning and statistical methods, namely Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Monte Carlo Simulation.

The goal is to determine the most accurate and reliable model for predicting future environmental conditions, and integrate this model into a web-based application to assist policymakers and industries in decision-making for climate adaptation and mitigation.

3. PROBLEM IDENTIFICATION:

Climate prediction remains a complex challenge due to the chaotic and interdependent nature of climatic systems. Existing methods often fall short in accurately forecasting specific variables like extreme temperature fluctuations, precipitation patterns, and other weather conditions on a localized scale. The challenge intensifies as conventional tools may not provide the level of granularity or predictive power necessary for effective adaptation planning. Additionally, the need for real-time, accessible tools that allow decision-makers to experiment with different scenarios remains largely unmet. This gap highlights the necessity for a dedicated platform like AdaptiPlan, which can utilize advanced forecasting models and provide users with accurate, data-rich predictions on climate patterns across varying time frames. This project responds to the pressing need for predictive tools that support data-driven climate adaptation strategies at both regional and community levels.

4. LITERATURE SURVEY:

Title	Inference
Prediction of Daily Climate Using Long Short-Term Memory (LSTM) Model [1]	The paper explores using Long Short-Term Memory (LSTM) networks to predict daily climate variables like temperature, leveraging historical data from Delhi of two years between 2013-2017. Traditional climate models are computationally intensive and may not capture local patterns well, but LSTM excels at processing time-dependent data. After preprocessing the data, LSTM model was trained, achieving good accuracy with a root mean squared error (RMSE) of 0.78. The results demonstrate the potential for LSTM networks to be applied to more complex climate variables and regions, offering more precise and localized climate forecasts in the future.
Use of the autoregressive integrated moving average (ARIMA) model to forecast near-term regional temperature and precipitation [2]	The paper explores ARIMA model, a statistical approach, to provide regional temperature and precipitation forecasts based on historical observations for 18 years. The ARIMA model was chosen due to its ability to account for temporal correlation and skewed distributions in climate data, improving forecasting accuracy over simpler methods like linear trends. The model was extended to estimate confidence intervals for temperature and precipitation extremes and simulate daily climate conditions. The predictions made by ARIMA model shows its reliability and adaptability for engineering applications, offering a robust alternative to traditional statistical forecasting methods.
A brief introduction to Monte Carlo simulation [3]	This paper gives introduction about Monte Carlo simulation on how it is becoming increasingly recognized for its powerful ability to manage uncertainty and variability in complex models. Unlike deterministic simulation, which offers a single outcome by assuming fixed parameters, Monte Carlo simulation explores a range of possible outcomes by incorporating random variations in the model's inputs. This method provides valuable insights into how different factors can influence results over time, making it a robust tool for understanding and predicting complex systems. Its versatility and effectiveness have made Monte Carlo simulation a staple in various fields, from finance to engineering, and it holds significant promise for enhancing decision-making processes. By capturing the full spectrum of potential scenarios, Monte Carlo simulation offers a deeper understanding of risk and uncertainty, which is increasingly important in today’s data-driven world.
LSTM-CM: a hybrid approach for natural drought prediction based on deep learning and climate models [4]	In this paper the hybrid LSTM-CM model developed in this study combines the strengths of both machine learning (LSTM-SA) and climate models (GS5) to improve drought prediction accuracy. By using the low bias of LSTM-SA and the strong physical simulation capabilities of GS5, LSTM-CM enhances drought forecasting, especially for 1-, 2-, and 3-month lead times, with improved skill scores ranging from 29.17% to 54.29%. This model shows better performance in detecting drought events with lower uncertainty than the stand-alone models, making it a promising tool for more reliable drought prediction.
Time series analysis of climate variables using seasonal ARIMA approach [5]	The research paper presents a new hybrid model, LSTM-CM, for predicting droughts by combining a long short-term memory (LSTM) model with a climate model (GloSea5). This model is compared against the standalone LSTM model (LSTM-SA) and the GloSea5 model (GS5) in terms of accuracy, bias, and overall performance. The results show that LSTM-CM outperforms both models by reducing bias, capturing the physical processes accurately, and improving

	prediction skill scores for 1-, 2-, and 3-month lead times. The LSTM-CM model effectively detects droughts with lower uncertainty, making it a reliable tool for drought forecasting.
Representing uncertainty in climate change scenarios: a Monte-Carlo approach [6]	The research paper presents a method for handling uncertainty in climate change scenarios using a Bayesian Monte-Carlo approach. This method accounts for uncertainties in future greenhouse gas emissions, climate sensitivity, and the limitations of climate models. The model uses various emissions scenarios and climate sensitivities to produce a range of possible future climate conditions. By applying these scenarios to impact models, such as those for hydrology, it provides a probability distribution of outcomes, helping decision-makers assess risks and plan for various future climate conditions. This approach enhances understanding and management of uncertainty in climate impact assessments.

5. EXISTING SYSTEM ISSUES:

While numerous tools exist for climate prediction, most are constrained by limitations such as insufficient data granularity, lack of scenario flexibility, or restricted access to advanced models for non-expert users. Many of these systems do not integrate a wide variety of prediction models, and some are accessible only to researchers with specialized knowledge in climate science. For instance, conventional weather forecasting models may lack adaptability, making them less suited for long-term scenario analysis or climate adaptation planning. Furthermore, existing systems often have limited web-based interfaces that are not accessible to policymakers and planners, who need a user-friendly platform to experiment with potential outcomes. These limitations underscore the need for a more accessible, robust tool like AdaptiPlan, which can address these gaps by offering a comprehensive, interactive, and user-oriented solution.

6. PROPOSED SYSTEM DESIGN:

The proposed design for AdaptiPlan involves a user-friendly web application that provides climate predictions based on multiple computational models. The design incorporates the LSTM, ARIMA, and Monte Carlo models to allow users to predict key climate variables like temperature, precipitation, and atmospheric pressure. The web interface will feature interactive modules where users can input different parameters, such as location coordinates (latitude and longitude) and forecast periods, to explore various climate scenarios. Behind the scenes, AdaptiPlan employs a pipeline that preprocesses ERA5 data, trains predictive models, and visualizes outputs in an accessible format. The modular architecture enables easy scaling, meaning new datasets and predictive models can be integrated as climate research progresses.

7. ALGORITHMS DISCUSSED:

- LSTM (Long Short-Term Memory): A deep learning model designed to handle time-series data by capturing both long-term and short-term dependencies. It's well-suited for modeling climate patterns due to its ability to learn complex temporal relationships.

Input:
Input sequence $X = [x_1, x_2, ..., x_t]$ where $x_t \in \mathbb{R}^n$ at time step t, T is the total length of the sequence.
Initial hidden state h_0, and cell state c_0.
Model parameters: input weight matrices W_i, W_f, W_o, W_c, recurrent weight matrices U_i, U_f, U_o, U_c, bias vectors b_i, b_f, b_o, b_c.

Output:
Output sequence $H = [h_1, h_2, ..., h_t]$, where each $h_t \in \mathbb{R}^m$, and final hidden and cell states h_t, c_t.
Step 1: Initialization
Initialize the hidden state h_0 and cell state c_0 to zero vectors: $h_0 = 0$, $c_0 = 0$.
Step 2: Iterate through the sequence
For each time step t = 1, 2, ..., T, repeat the following steps:
Step 2.1: Forget gate computation
Compute the forget gate vector f_t, which decides what information from the previous cell state to forget:
$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$
Step 2.2: Input gate computation
Compute the input gate vector i_t, which decides what new information to store in the cell state:
$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$
Step 2.3: Candidate cell state
Compute the candidate cell state $\tilde{c}_t$, which is the potential update for the cell state:
$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$
Step 2.4: Update cell state
Update the cell state c_t using the forget gate and the input gate:
$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$
Step 2.5: Output gate computation
Compute the output gate vector o_t, which decides the next hidden state:
$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$
Step 2.6: Update hidden state
Update the hidden state h_t based on the cell state and the output gate:
$h_t = o_t \odot \tanh(c_t)$
Step 3: Final Output
After iterating through all time steps t = 1, 2, ..., T, the final hidden state h_t and cell state c_t are obtained. The output sequence $H = [h_1, h_2, ..., h_t]$ represents the final hidden states at each time step.
Step 4: Prediction
For tasks like sequence prediction, apply a fully connected layer to the final hidden state h_t (or each h_t) to map it to the desired output dimension.

- ARIMA (Autoregressive Integrated Moving Average): A classical statistical model used for time-series forecasting. ARIMA captures trends and patterns over time, making it suitable for forecasting climate data with temporal correlations.

Input:

Time series data $X = [x_1, x_2, ..., x_t]$ where x_t is the value at time step t.
Parameters p, d, and q:
  - p: Order of the autoregressive (AR) model.
  - d: Degree of differencing.
  - q: Order of the moving average (MA) model.
Output:
Forecasted values $\hat{x}_{t+1}, \hat{x}_{t+2}, ...$
Step 1: Differencing

Apply differencing d times to the time series X to make it stationary:
$Y_t = \Delta^d X_t$
where Δ represents differencing.
Step 2: AR Model

Fit an AR model of order p:
$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + ... + \varphi_p Y_{t-p} + \varepsilon_t$
where ε_t is the error term.
Step 3: MA Model

Fit an MA model of order q:
$\varepsilon_t = \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q} + Z_t$
where Z_t is white noise.
Step 4: Combining AR and MA

Combine the AR and MA components to form the ARIMA model:
$\hat{x}_t = \varphi_1 \hat{x}_{t-1} + ... + \varphi_p \hat{x}_{t-p} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q}$
Step 5: Forecasting

Use the fitted ARIMA model to forecast future values.

- Monte Carlo Simulation: A probabilistic model that simulates a wide range of possible outcomes by incorporating random inputs. It's useful for assessing uncertainty in climate predictions by running multiple scenarios.

Input:
Probability distribution P of possible outcomes.
Number of simulations N.
Output:
Estimated outcome based on the average of simulations.
Step 1: Initialize

Set the number of simulations N and define the probability distribution P.
Step 2: Simulate

For each simulation i = 1, 2, ..., N:
  - Draw a random sample x_i from the distribution P.
  - Compute the outcome y_i based on x_i.
Step 3: Aggregate Results

Calculate the mean of all simulated outcomes:
$\hat{y} = (1/N) \Sigma_{i}^{n}=_1 y_i$
Step 4: Output

Return the estimated outcome $\hat{y}$.

8. UML DIAGRAM:

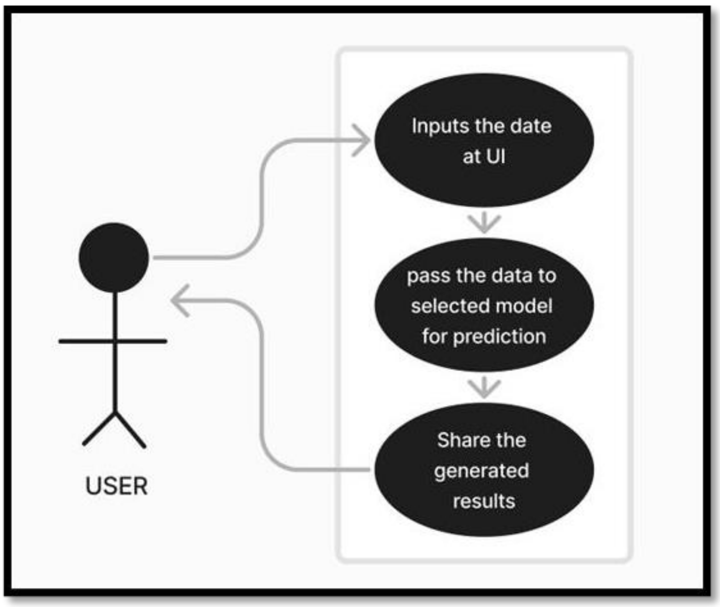
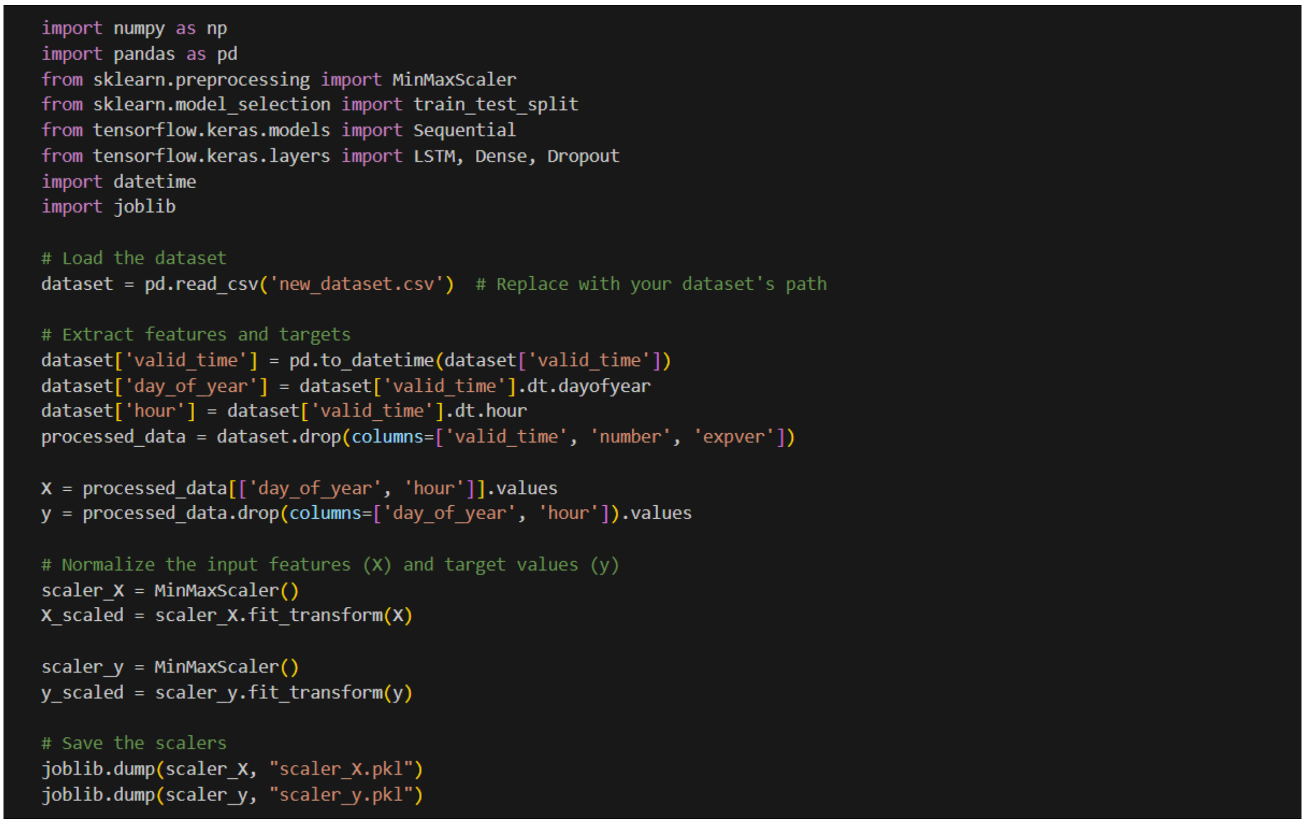

Inputs the date at UI
pass the data to selected model for prediction
Share the generated results
USER

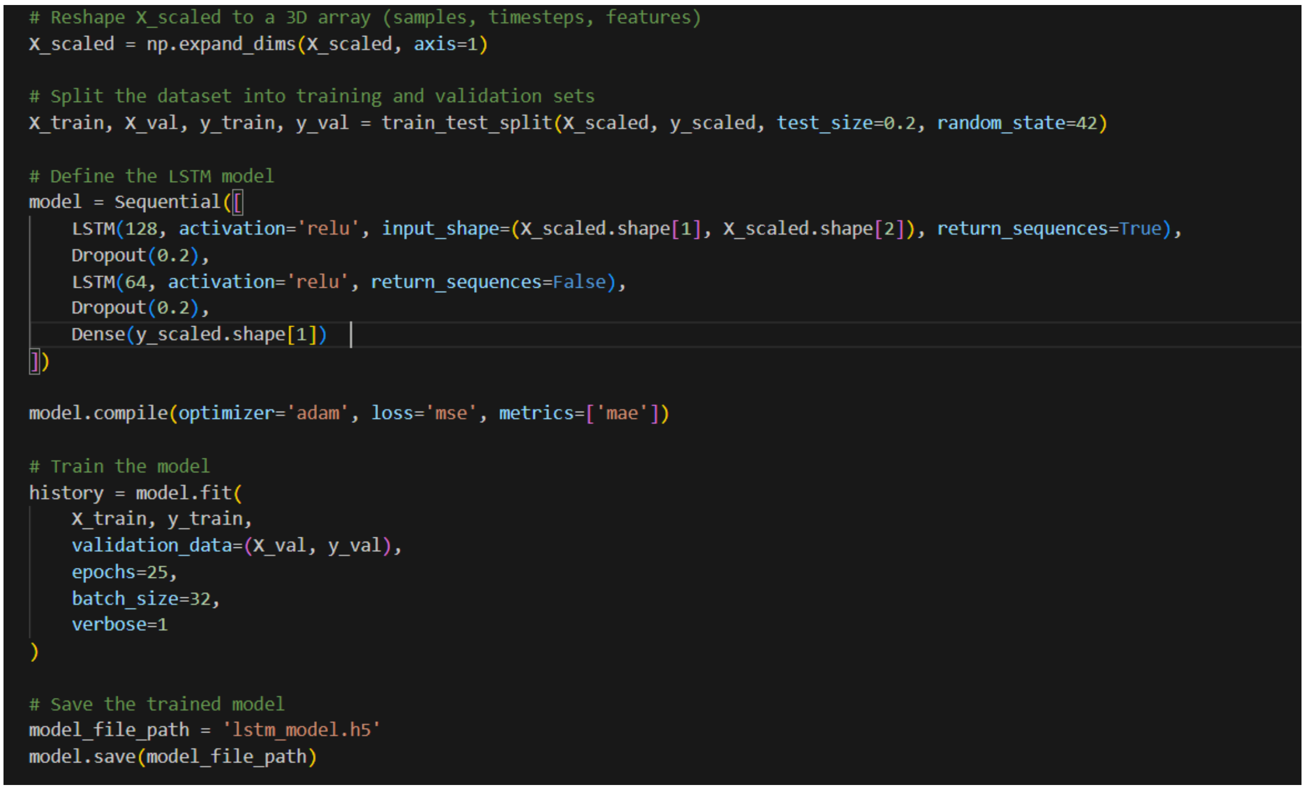
Fig 2.2. UML

9. RESULT AND DISCUSSIONS:

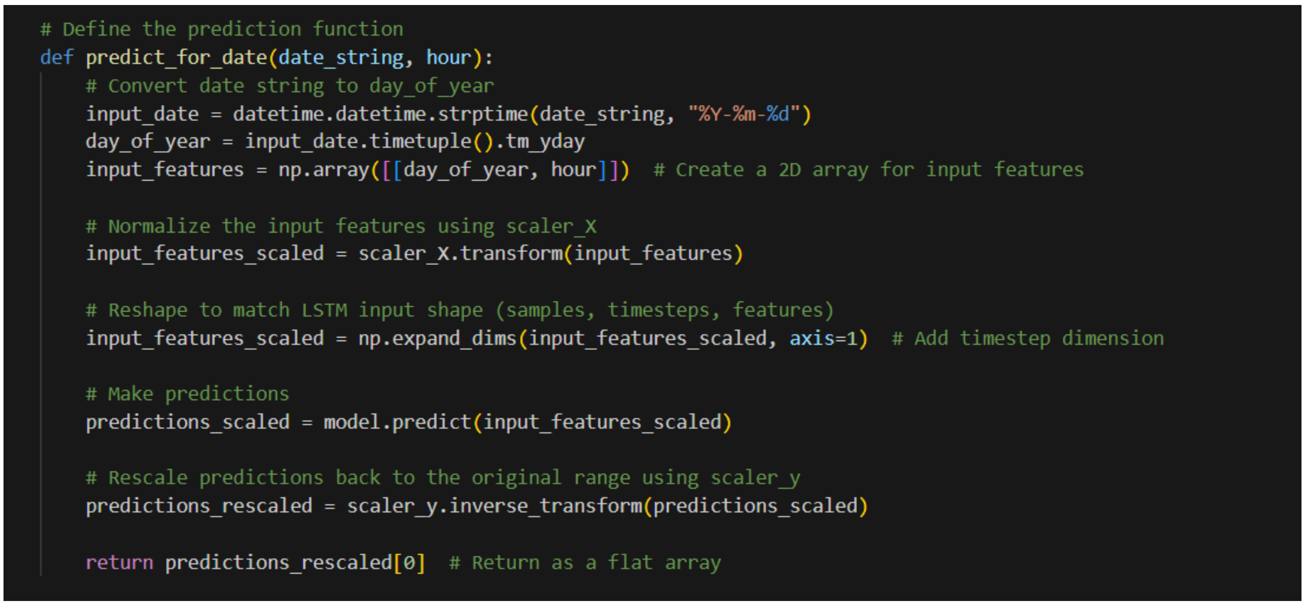
Models	**Architecture**	**Computational Complexity**	**MSE for Event's Prediction**
LSTM	Recurrent Neural Network with Long Short-Term Memory units	Moderate, due to sequential dependencies and time steps	It gave the best results.
ARIMA	Statistical model (AutoRegressive Integrated Moving Average)	Lower, but can be computationally expensive with high-order models	It performed better than Monte Carlo but worse than LSTM
Monte Carlo Simulation	Probabilistic model based on random sampling and repeated experiments	Depends on the number of simulations (can be high)	It was not good comparing other two models.



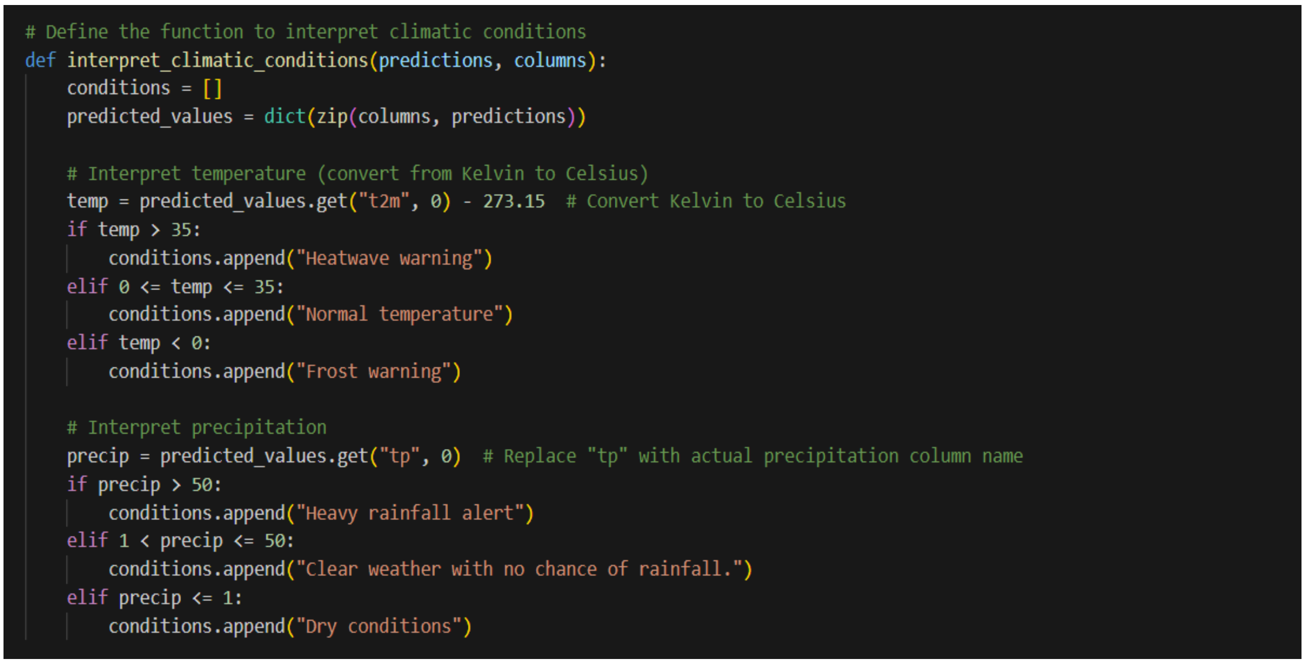
```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import datetime
import joblib

# Load the dataset
dataset = pd.read_csv('new_dataset.csv')  # Replace with your dataset's path

# Extract features and targets
dataset['valid_time'] = pd.to_datetime(dataset['valid_time'])
dataset['day_of_year'] = dataset['valid_time'].dt.dayofyear
dataset['hour'] = dataset['valid_time'].dt.hour
processed_data = dataset.drop(columns=['valid_time', 'number', 'expver'])

X = processed_data[['day_of_year', 'hour']].values
y = processed_data.drop(columns=['day_of_year', 'hour']).values

# Normalize the input features (X) and target values (y)
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)

scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(y)

# Save the scalers
joblib.dump(scaler_X, "scaler_X.pkl")
joblib.dump(scaler_y, "scaler_y.pkl")
```



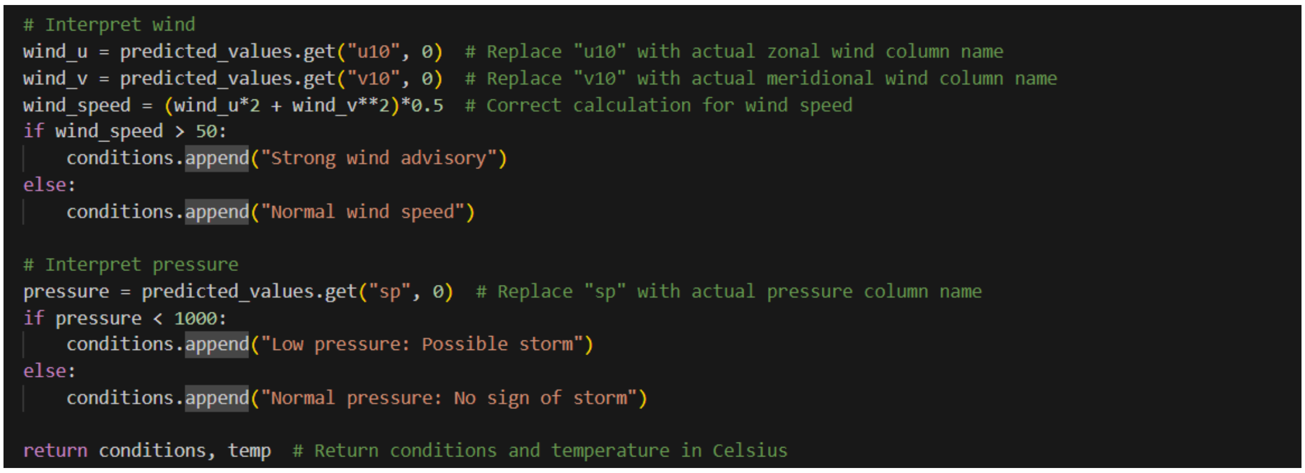
```python
# Reshape X_scaled to a 3D array (samples, timesteps, features)
X_scaled = np.expand_dims(X_scaled, axis=1)

# Split the dataset into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_scaled, y_scaled, test_size=0.2, random_state=42)

# Define the LSTM model
model = Sequential([
    LSTM(128, activation='relu', input_shape=(X_scaled.shape[1], X_scaled.shape[2]), return_sequences=True),
    Dropout(0.2),
    LSTM(64, activation='relu', return_sequences=False),
    Dropout(0.2),
    Dense(y_scaled.shape[1])
])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Train the model
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=25,
    batch_size=32,
    verbose=1
)

# Save the trained model
model_file_path = 'lstm_model.h5'
model.save(model_file_path)
```



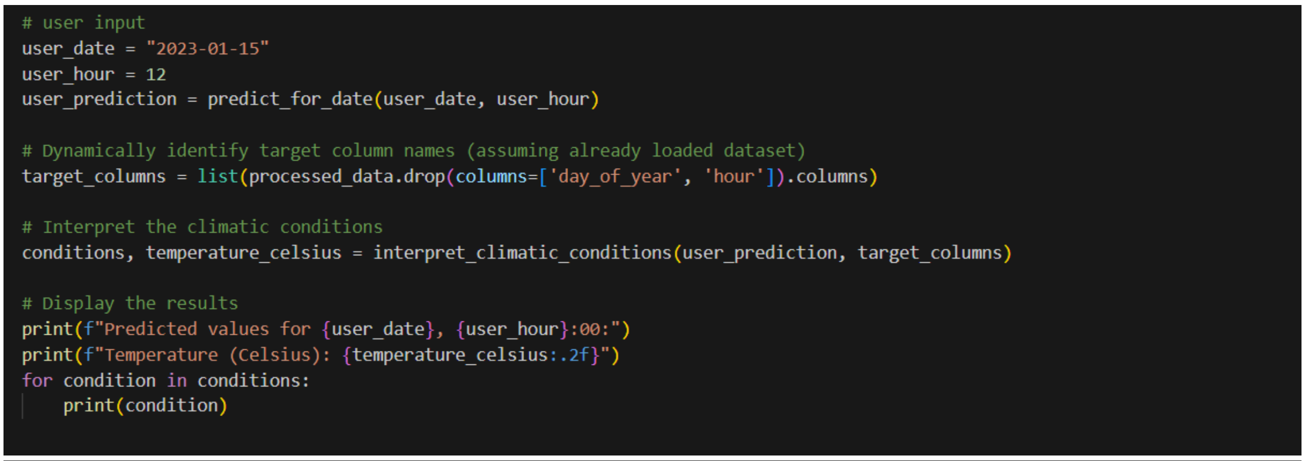
```python
# Define the prediction function
def predict_for_date(date_string, hour):
    # Convert date string to day_of_year
    input_date = datetime.datetime.strptime(date_string, "%Y-%m-%d")
    day_of_year = input_date.timetuple().tm_yday
    input_features = np.array([[day_of_year, hour]])  # Create a 2D array for input features

    # Normalize the input features using scaler_X
    input_features_scaled = scaler_X.transform(input_features)

    # Reshape to match LSTM input shape (samples, timesteps, features)
    input_features_scaled = np.expand_dims(input_features_scaled, axis=1)  # Add timestep dimension

    # Make predictions
    predictions_scaled = model.predict(input_features_scaled)

    # Rescale predictions back to the original range using scaler_y
    predictions_rescaled = scaler_y.inverse_transform(predictions_scaled)

    return predictions_rescaled[0]  # Return as a flat array
```



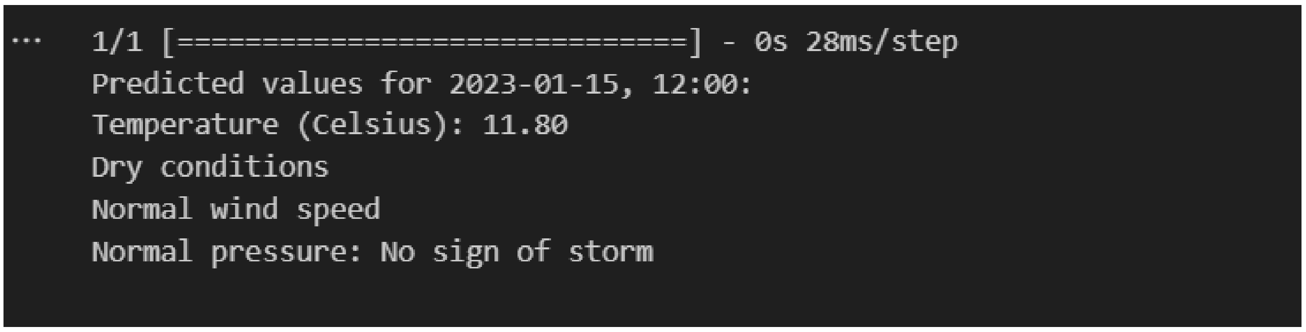
```python
# Define the function to interpret climatic conditions
def interpret_climatic_conditions(predictions, columns):
    conditions = []
    predicted_values = dict(zip(columns, predictions))

    # Interpret temperature (convert from Kelvin to Celsius)
    temp = predicted_values.get("t2m", 0) - 273.15  # Convert Kelvin to Celsius
    if temp > 35:
        conditions.append("Heatwave warning")
    elif 0 <= temp <= 35:
        conditions.append("Normal temperature")
    elif temp < 0:
        conditions.append("Frost warning")

    # Interpret precipitation
    precip = predicted_values.get("tp", 0)  # Replace "tp" with actual precipitation column name
    if precip > 50:
        conditions.append("Heavy rainfall alert")
    elif 1 < precip <= 50:
        conditions.append("Clear weather with no chance of rainfall.")
    elif precip <= 1:
        conditions.append("Dry conditions")
```



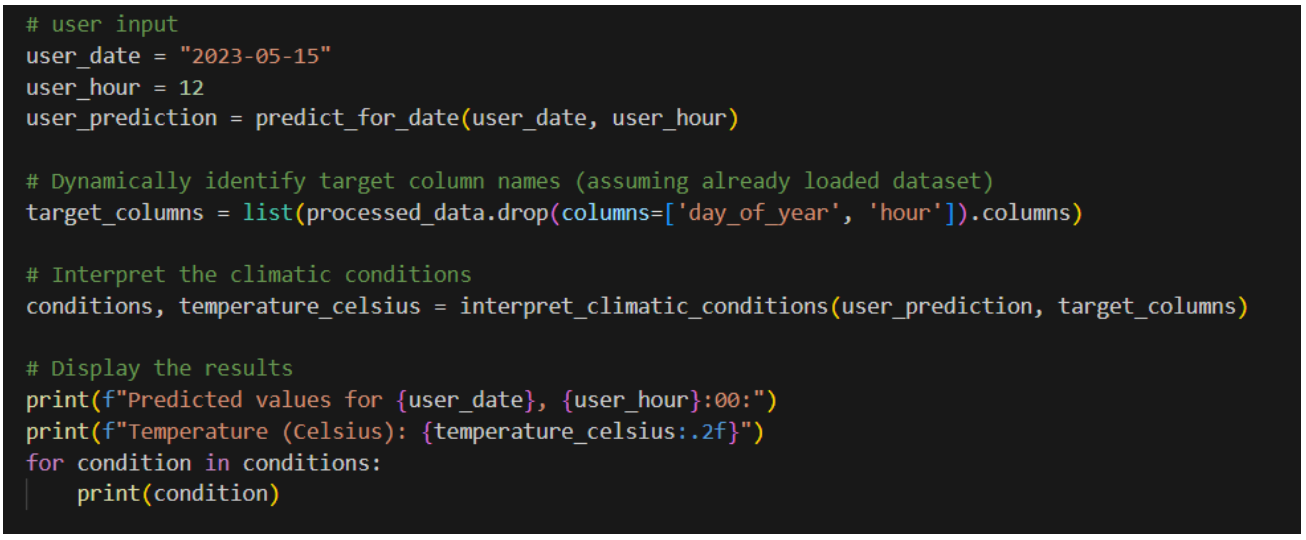
```python
# Interpret wind
wind_u = predicted_values.get("u10", 0)  # Replace "u10" with actual zonal wind column name
wind_v = predicted_values.get("v10", 0)  # Replace "v10" with actual meridional wind column name
wind_speed = (wind_u*2 + wind_v**2)*0.5  # Correct calculation for wind speed
if wind_speed > 50:
    conditions.append("Strong wind advisory")
else:
    conditions.append("Normal wind speed")

# Interpret pressure
pressure = predicted_values.get("sp", 0)  # Replace "sp" with actual pressure column name
if pressure < 1000:
    conditions.append("Low pressure: Possible storm")
else:
    conditions.append("Normal pressure: No sign of storm")

return conditions, temp  # Return conditions and temperature in Celsius
```



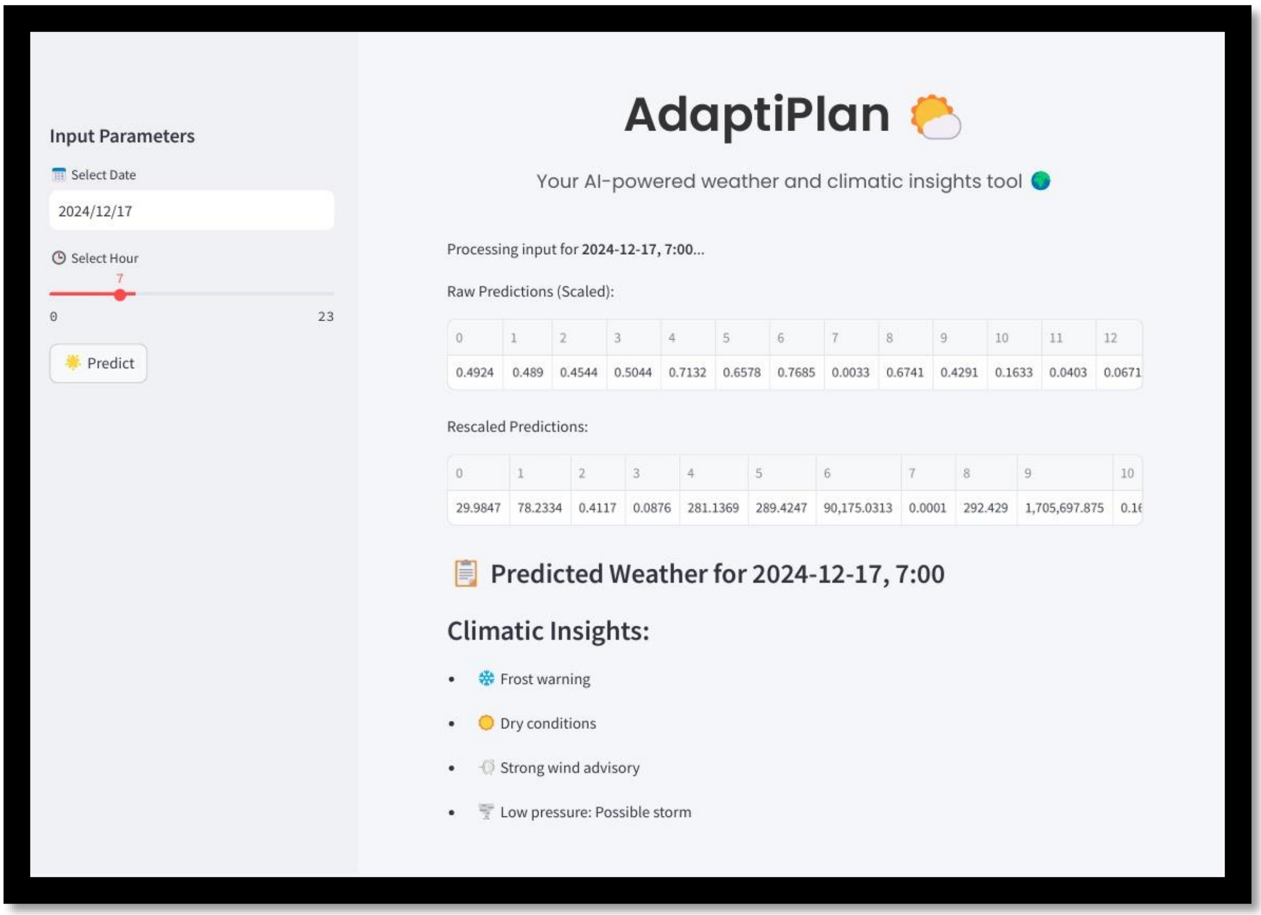
```python
# user input
user_date = "2023-01-15"
user_hour = 12
user_prediction = predict_for_date(user_date, user_hour)

# Dynamically identify target column names (assuming already loaded dataset)
target_columns = list(processed_data.drop(columns=['day_of_year', 'hour']).columns)

# Interpret the climatic conditions
conditions, temperature_celsius = interpret_climatic_conditions(user_prediction, target_columns)

# Display the results
print(f"Predicted values for {user_date}, {user_hour}:00:")
print(f"Temperature (Celsius): {temperature_celsius:.2f}")
for condition in conditions:
    print(condition)
```



```
... 1/1 [==============================] - 0s 28ms/step
    Predicted values for 2023-01-15, 12:00:
    Temperature (Celsius): 11.80
    Dry conditions
    Normal wind speed
    Normal pressure: No sign of storm
```



```python
# user input
user_date = "2023-05-15"
user_hour = 12
user_prediction = predict_for_date(user_date, user_hour)

# Dynamically identify target column names (assuming already loaded dataset)
target_columns = list(processed_data.drop(columns=['day_of_year', 'hour']).columns)

# Interpret the climatic conditions
conditions, temperature_celsius = interpret_climatic_conditions(user_prediction, target_columns)

# Display the results
print(f"Predicted values for {user_date}, {user_hour}:00:")
print(f"Temperature (Celsius): {temperature_celsius:.2f}")
for condition in conditions:
    print(condition)
```



AdaptiPlan 🌤️

Your AI-powered weather and climatic insights tool 🌍

Processing input for **2024-12-17, 7:00...**

Raw Predictions (Scaled):

0	1	2	3	4	5	6	7	8	9	10	11	12
0.4924	0.489	0.4544	0.5044	0.7132	0.6578	0.7685	0.0033	0.6741	0.4291	0.1633	0.0403	0.0671

Rescaled Predictions:

0	1	2	3	4	5	6	7	8	9	10
29.9847	78.2334	0.4117	0.0876	281.1369	289.4247	90,175.0313	0.0001	292.429	1,705,697.875	0.1

📋 Predicted Weather for 2024-12-17, 7:00

Climatic Insights:

- ❄️ Frost warning
- 🟡 Dry conditions
- 💨 Strong wind advisory
- 🌪️ Low pressure: Possible storm

10. CONCLUSION:

In conclusion, AdaptiPlan represents a significant step forward in climate adaptation tools, providing stakeholders with the ability to explore and plan for potential climate impacts. By integrating LSTM, ARIMA, and Monte Carlo models with the ERA5 dataset, the application offers a versatile and accessible solution for scenario modeling. This project contributes to the field of climate science by offering a tool that combines predictive accuracy with user-friendly access, facilitating evidence-based decision-making for climate resilience.

11. FUTURE WORK:

Future developments for AdaptiPlan could involve expanding the model library to include other deep learning architectures such as Transformer models, which have shown promise in time-series analysis. Additionally, integrating real-time data inputs could further enhance the tool's accuracy and applicability. A future version of AdaptiPlan might also incorporate a feedback system, where user insights and local climate data could continuously improve the prediction models. Finally, potential collaborations with environmental agencies and researchers could transform AdaptiPlan into a widely adopted resource in climate adaptation efforts.

12. REFERENCES:

[1] Xu, J., Wang, Z., Li, X., Li, Z., & Li, Z. (2024). Prediction of Daily Climate Using Long Short-Term Memory (LSTM) Model. International Journal of Innovative Science and Research Technology, 83-90

[2] Lai, Y., & Dzombak, D. A. (2020). Use of the autoregressive integrated moving average (ARIMA) model to forecast near-term regional temperature and precipitation. Weather and forecasting, 35(3), 959-976.

[3] Bonate, P. L. (2001). A brief introduction to Monte Carlo simulation. Clinical pharmacokinetics, 40, 15-22.

[4] Vo, T. Q., Kim, S. H., Nguyen, D. H., & Bae, D. H. (2023). LSTM-CM: a hybrid approach for natural drought prediction based on deep learning and climate models. *Stochastic Environmental Research and Risk Assessment*, *37*(6), 2035-2051.

[5] Dimri, T., Ahmad, S., & Sharif, M. (2020). Time series analysis of climate variables using seasonal ARIMA approach. *Journal of Earth System Science*, *129*, 1-16.

[6] New, M., & Hulme, M. (2000). Representing uncertainty in climate change scenarios: a Monte-Carlo approach. *Integrated assessment*, *1*(3), 203-213.