BENNY BEAVER'S BODEMENT

Members: Alona Rudchenko, Michael Waldrop, Jordan Bleck, and Biran Shah

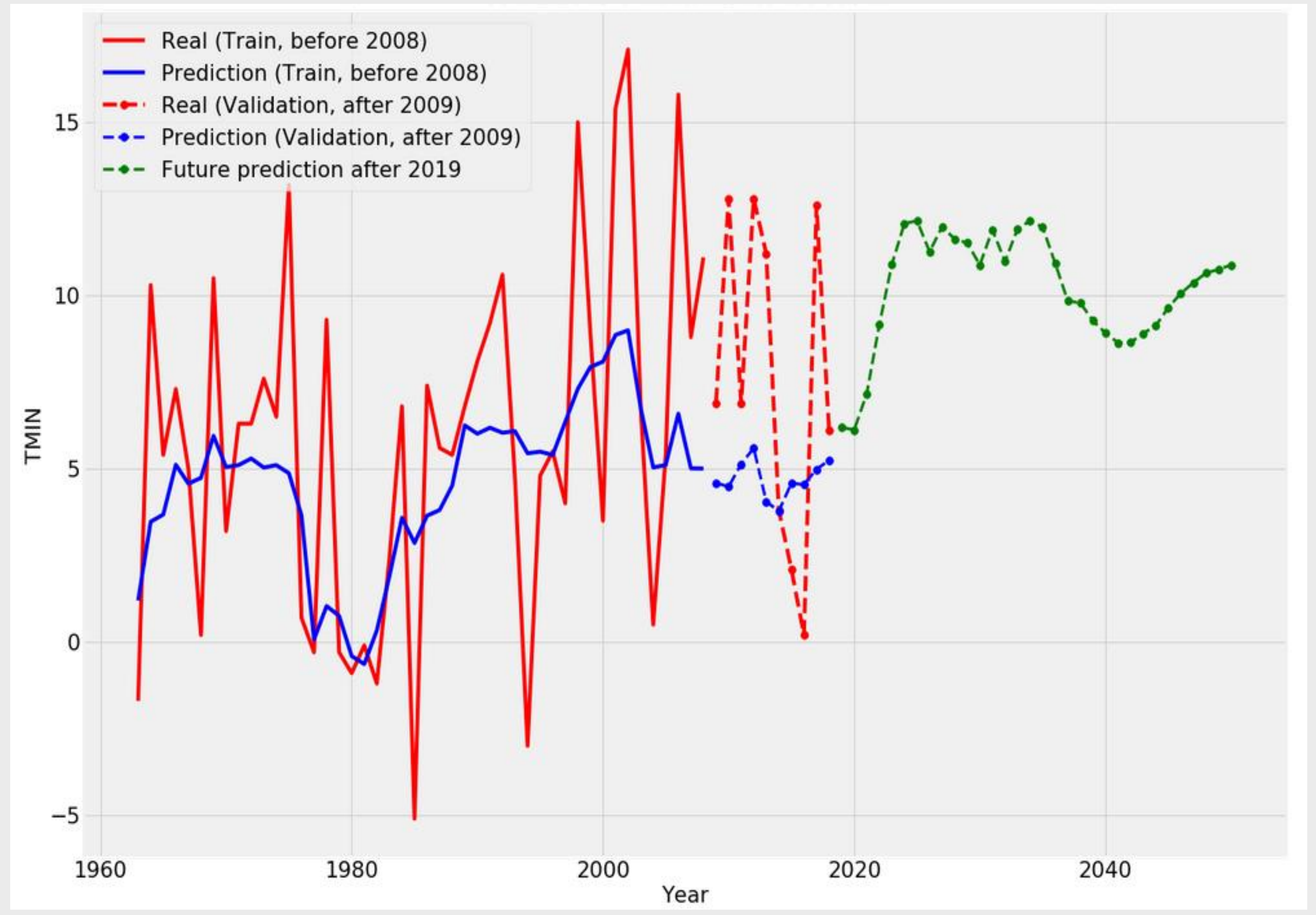
Long term temperature forecasting for major global cities with a local weather station.

TEMPERATURE FORECAST AND WEATHER STATION

- Our program focuses on two primary components. First, analyzing historical data sets of major global cities to forecast the annual mean temperature in major global cities, and presenting these results along with interactive global climate graphs on our website. The second, running a local weather station based on the user's location via a Raspberry Pi.
- Our program was written in the Python3 programming language on either the OSU School Server or Jupyter Notebook.
- Our data sources for all cities' historical datasets comes from either the US National Oceanic and Atmospheric Administration or Visual Crossing.
- Common libraries and frameworks include the following:
  - Flask
  - jinja2
  - Numpy
  - Scipy
  - Matplotlib
  - SQLite
  - Pandas
  - Keras
  - Tensorflow

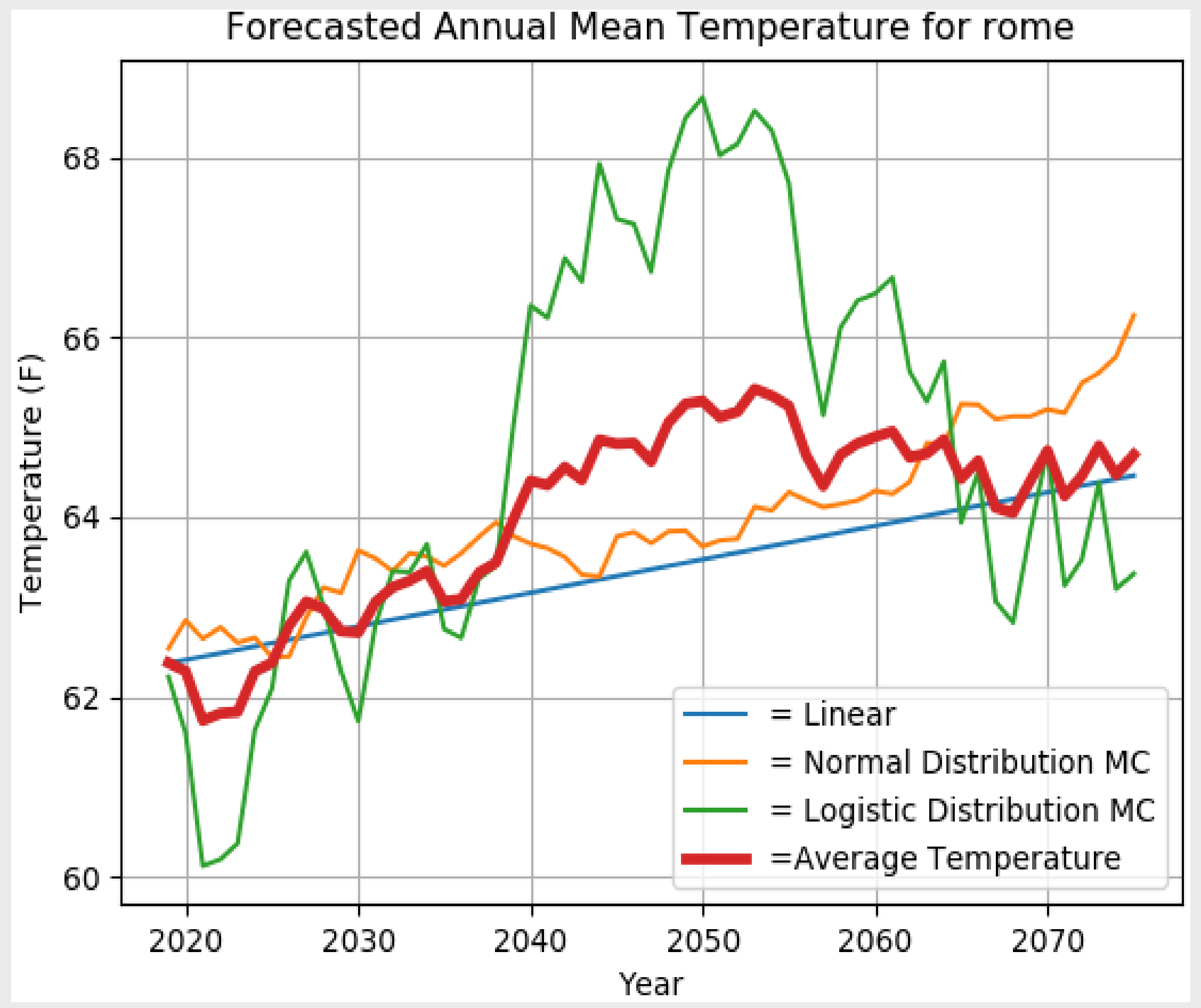
Oregon State University

Machine Learning based Forecasting

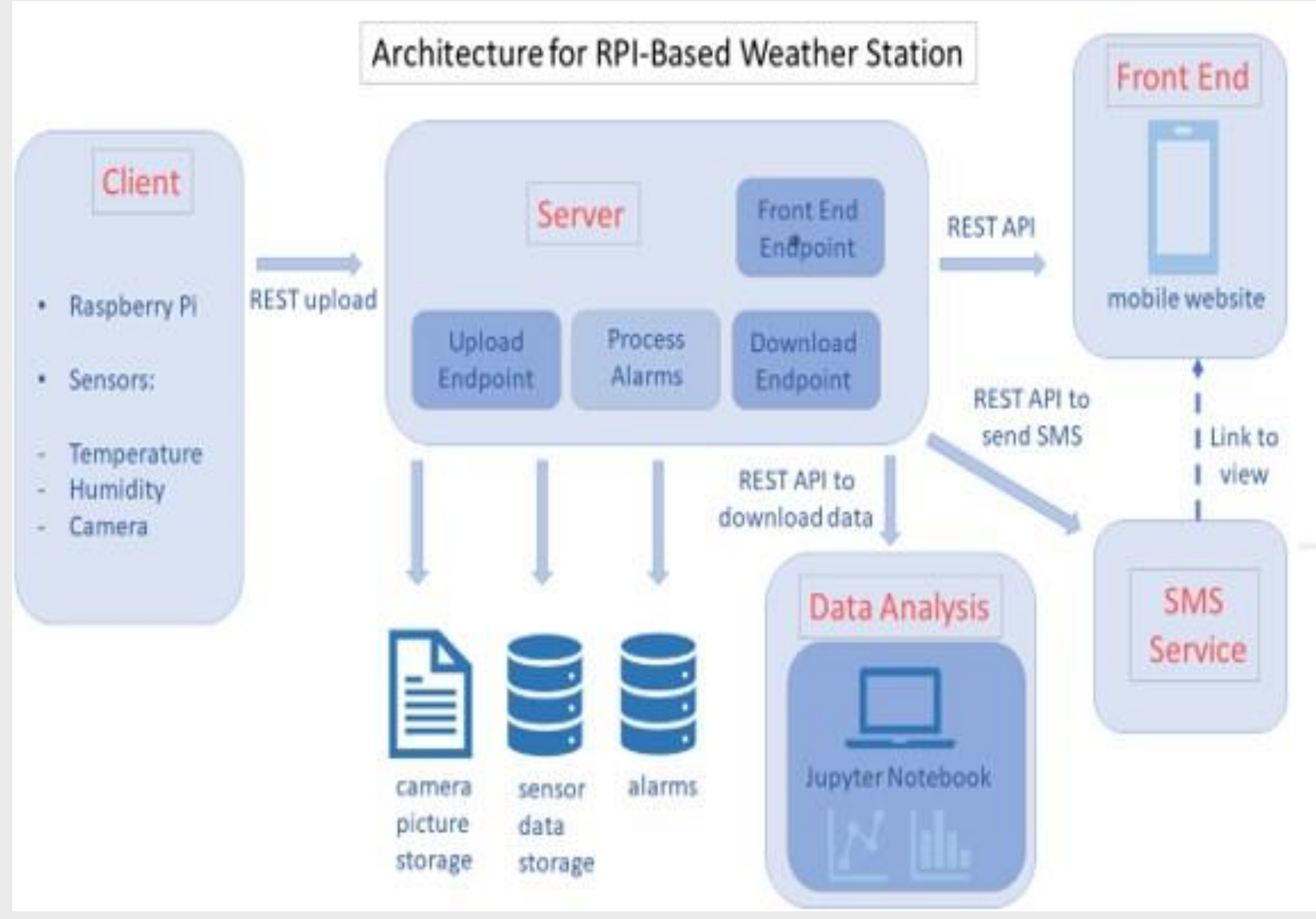


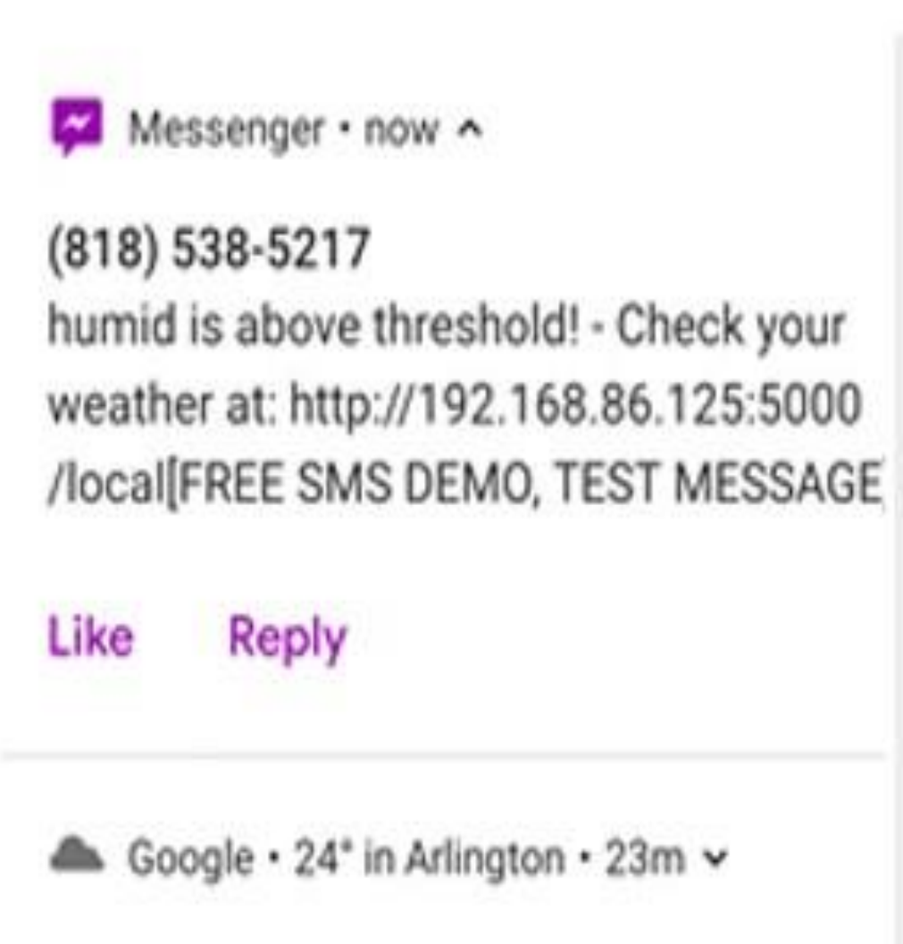
- A graph of the machine learning model for forecasting the minimum temperature of New York City up to the year 2050 with training and validation models and using an LTSM with attention layer for city weather data towards increasing accuracy.
- This utilizes a twenty year look back function that carefully considers the past two decades of weather data given a starting and ending point.

Probability based Forecasting



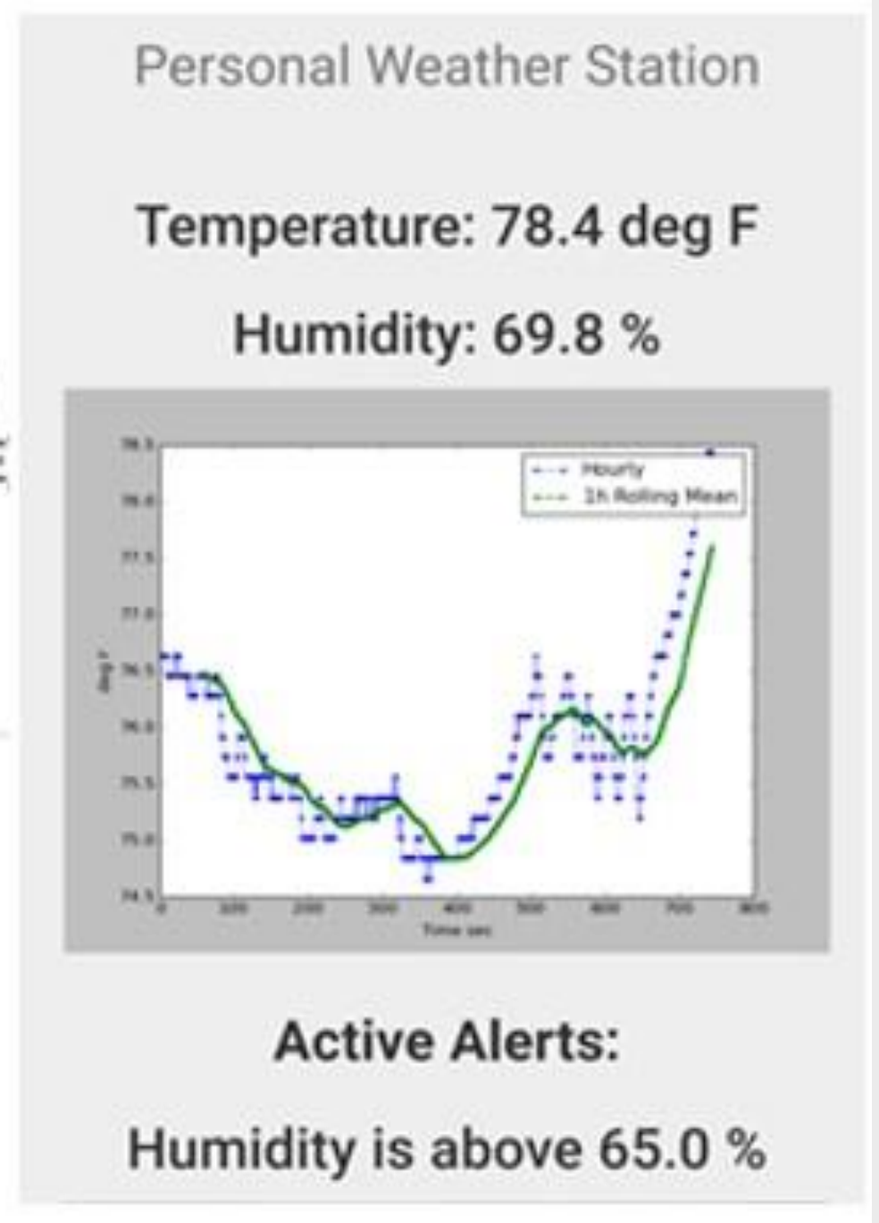
- Annual mean temperature forecast for Rome.
- The simulations required a simple and weighted moving averages for the mean annual temperatures.
- The simulations were a modified version of a Monte Carlo Simulation that preserved the fundamental random sampling produced forecasts via a normal and a logistical alternative parameterization distributions, along with a linear model.




Messenger • now

(818) 538-5217

humid is above threshold! - Check your weather at: http://192.168.86.125:5000/local[FREE SMS DEMO, TEST MESSAGE


Personal Weather Station

Temperature: 78.4 deg F

Humidity: 69.8 %

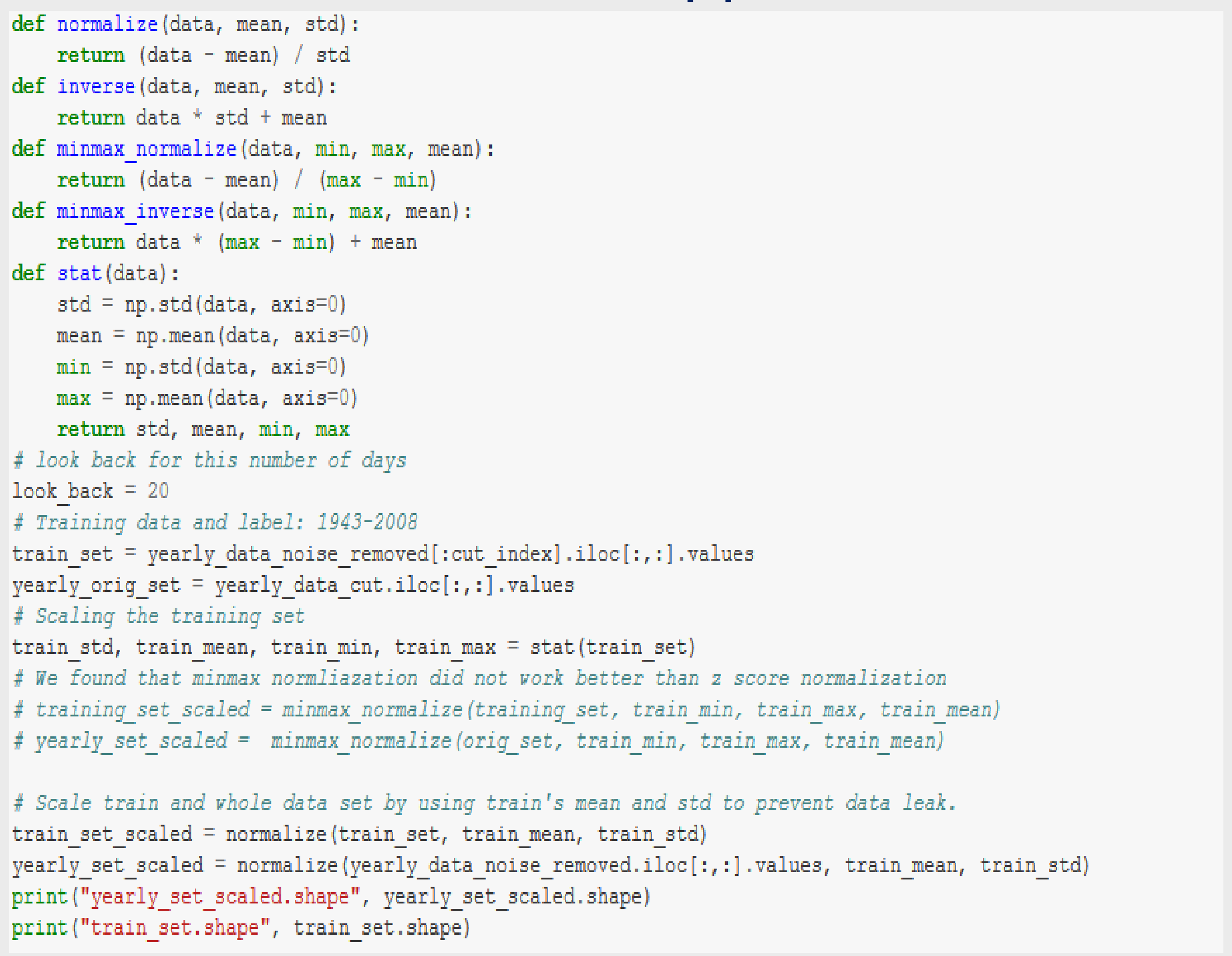
Active Alerts:

Humidity is above 65.0 %

Local Weather Station

- Raspberry Pi (RPI)-based weather station collects weather data: temperature, humidity, and weather images, recorded every minute. All collected data can be downloaded in csv format.
- User can set alerts to receive SMS messages when measurements are above/below defined threshold. The received SMS includes a URL to view current weather on a webpage directly on mobile device.
- Current weather data are shown on a webpage along with a webcam image and a graph of temperature over the last 24 hours with the rolling mean.

Code Snippet

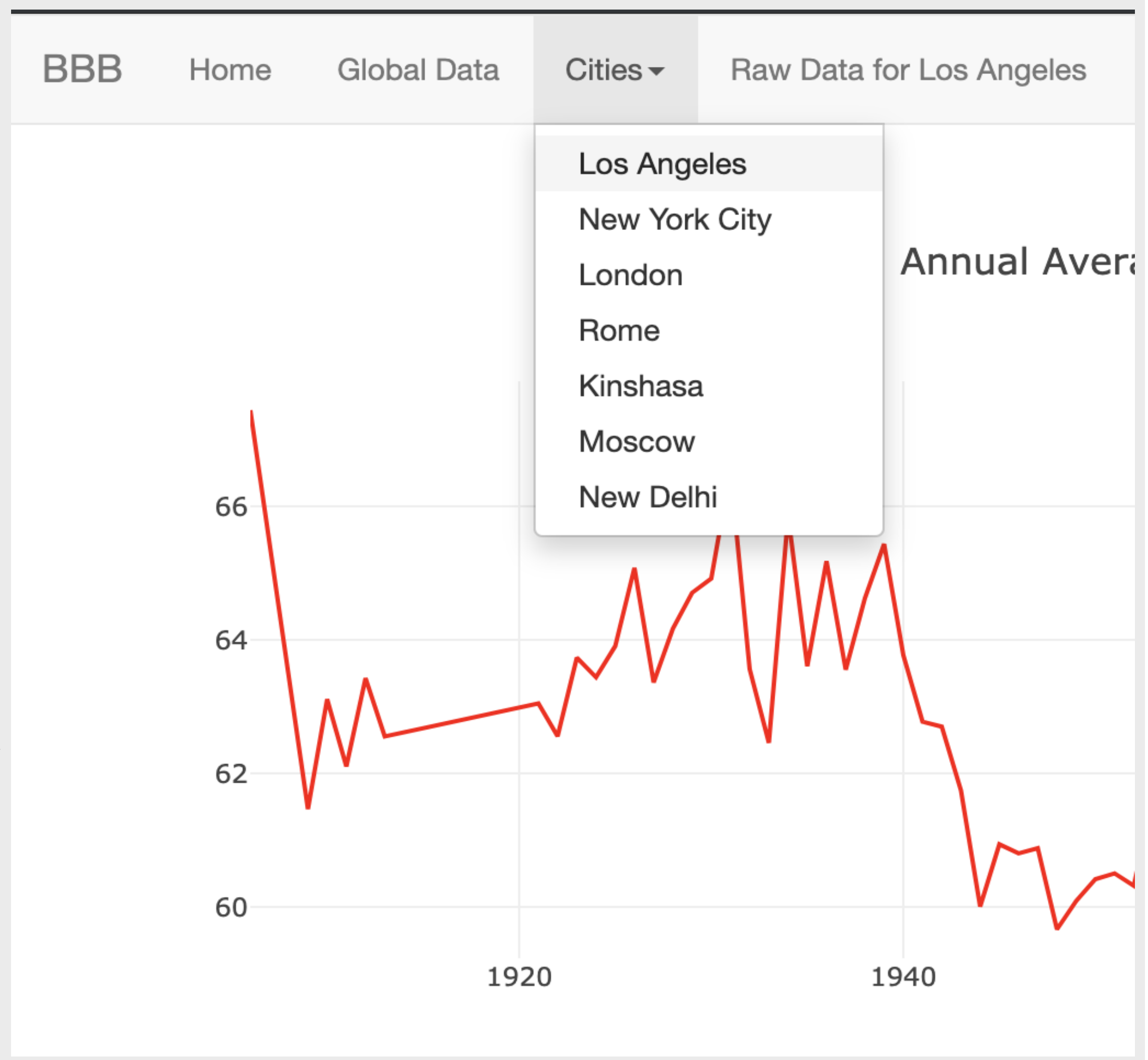


```python
def normalize(data, mean, std):
    return (data - mean) / std
def inverse(data, mean, std):
    return data * std + mean
def minmax_normalize(data, min, max, mean):
    return (data - mean) / (max - min)
def minmax_inverse(data, min, max, mean):
    return data * (max - min) + mean
def stat(data):
    std = np.std(data, axis=0)
    mean = np.mean(data, axis=0)
    min = np.std(data, axis=0)
    max = np.mean(data, axis=0)
    return std, mean, min, max
# look back for this number of days
look_back = 20
# Training data and label: 1943-2008
train_set = yearly_data_noise_removed[:cut_index].iloc[:,:].values
yearly_orig_set = yearly_data_cut.iloc[:,:].values
# Scaling the training set
train_std, train_mean, train_min, train_max = stat(train_set)
# We found that minmax normliazation did not work better than z score normalization
# training_set_scaled = minmax_normalize(training_set, train_min, train_max, train_mean)
# yearly_set_scaled =  minmax_normalize(orig_set, train_min, train_max, train_mean)

# Scale train and whole data set by using train's mean and std to prevent data leak.
train_set_scaled = normalize(train_set, train_mean, train_std)
yearly_set_scaled = normalize(yearly_data_noise_removed.iloc[:,:].values, train_mean, train_std)
print("yearly_set_scaled.shape", yearly_set_scaled.shape)
print("train_set.shape", train_set.shape)
```

- This sample code is from the machine learning algorithm.
- By looking back for a number of years while training the prediction model, we are able to track recent temperature trends. This look back number could be adjusted if necessary, but climate trends may not be consistent with larger time periods, especially with recent extreme climate events.

Program Website and Global Data



- Our website features a front and back end implementation for our global datasets studies including number of annual cyclones, average global annual temperature, precipitation, and sea-level.
- A user can upload historical data of their choice of city and interact with it in visual displays.
- The interactive display calculates in real time so database updates are available to the user.
- Back end flask applications allow easy app access and database controls.