CIS4930 Term Project

Miles Brosz,

Natalee Sama,

Aidan Mahoney,

Tyler Zuluaga

Florida State University

CIS4930; Special Topics

Prof. Mithila

4/11/25

Climate Change Impact Analyzer Python Project

This project pools data from the “NOAA National Centers for Environmental Information (NCEI)” weather API, in order to successfully display, predict, and normalize precipitation data across the span of 5 years (inclusive of 2020 - 2024) over three different Florida stations (Miami, Orlando, Tallahassee). We present such data in mediums- Scatter plots, box plots and an animated line graph. The Project is separated into main files: “algorithms.py”; which fittingly sees implementation of the algorithms we use for data prediction and detecting anomalies/miscellaneous skews from the data trend; “DataCollection.py”, being responsible for making the API request and sending the data to “/data/x.json”, ‘x’ being the corresponding station’s location; “cli.py”, our command-line interface, which parses command-line inputs and loads and processes the data in the aforementioned “climate_data”, alongside running the code for “visualizer.py” which interprets and displays the data in such called graphs. Extra functionality is also included with the HTML presentation as an optional alternative to the command-line interface, displays and navigation were made using Flask.

Functionality/Algorithms

Before we make any graphical interpretations of our data, we first needed to extract the dates where no precipitation occurs, being that we are pooling data across the entire year, over a number of years, this would skew the data drastically. The next step would be to normalize - in this, we format the precipitation ranges into easier to handle numbers such that they are floating point values in a range from zero to one; Zero, representing the minimum precipitation

value from our data set, and one the maximum. Normalization is important for training the learning model to accurately predict precipitation values for years outside of our sample size.

The learning model takes normalized data and iterates through the linear regression prediction formula. During each iteration, it updates a value weight, based on the error or the difference between the predicted value and the actual. The bias also works in the same manner. It then sets these to a new value, being its previous – (the updated value * learning rate). This allows the weight and bias in the next iteration to produce a more accurate prediction. The "Predict" function calculates the predicated precipitation, given a year, utilizing the final updated weight and bias from the regression training. We chose this style of learning model because we found the Linear-Regression idea that it is based from to be the most intuitive and seamless to implement.

Our anomaly algorithm takes the normalized precipitation values from the "data_processor.py" and feeds it to the boxplot in the matplotlib library. In our main interface file, we make a DataProcessor instance for each of our data sets so that we can have these normalized values separated per location and can display them individually for easier interpretation. We make a subplot from each of these data sets and attach them to their appropriate labels. Only the anomalies are shown as dots outside of the normal data distribution. The attributes for each of the plots are calculated by the boxplot's internal functionality.

Upon reaching the final functional task of clustering, in needing to group stations of similar climates/ precipitation amounts, we saw it fit to pool other kinds of data such as temperature, dryness and/or wetness, as these could contribute to precipitation. We calculated

these values and within a certain threshold were able to cluster various stations based on these attributes, alongside the amount of precipitation. We then display this in a multidimensional graph of intensity of rain and intensity of temperature.

We also included other supplemental functionalities in the form of an HTML overlay to interact with the same functionalities as the command-line interface, as well as an animated graph to view the prediction values of our learning model, compared to the actual precipitation values. It should be noted that, due to the large size of our data collection, the animated graph functions at a slow rate.