CatastraSense: A Weather Information and Prediction Program

MAD2502: Introduction to Computational Mathematics

Presentation Link: https://youtu.be/bxExqauOhsk

Shravya Sama and Tatum Bowen

Due: 12.11.23

Introduction

CatastraSense is designed to inform users about historical and potential future weather data, specifically related to the Florida area. There are a large number of websites and apps related to weather and there are even more databases, but this program is made to be user friendly. Rather than having to look at a large amount of data to gain insight to past weather or climate patterns, we have created a way for the user to input a date for one to two cities and then any combination of maximum temperature, minimum temperature, mean temperature, or precipitation can be provided for said date. Furthermore, we have used AI through the sklearn package to be able to predict whether a natural disaster will occur on a particular day in the future using information about what days in the past natural disasters have occurred on.

Previous Work

To start, our code was an original idea and there was no original base code that we used to add on to or modify; however, we did use some segments of code to aid in some of the coding aspects that we did not yet have experience in. As an overview, we used youtube videos (GitHub code explanations), websites for some of the packages, and ChatGPT. The packages were used to shorten the code and make some of the elements that were more difficult to code a little bit easier. The videos and GitHub explanations were used to help us understand a methodology, but none of the code was explicitly copied and added to our code. Lastly, ChatGPT was used primarily for minor error corrections and making sure the code was PEP 8 compliant and uniform.

Going through the code, this is an elaboration of where specific sources were used and what their purpose was in reaching our goals for this code. In the first section of the code, we had

data cleaning and in order to figure out how to replace empty values and fill the holes, we asked Chat GPT for different methods and decided to use the '.fillna'. In the next section, we merged all of our datasets and, in order to make sure we did so correctly, we used the methodology demonstrated in a GeeksforGeeks webpage. For the final data processing and completion step, ChatGPT was used to organize the cities and format them to be appended to the data— though it should be noted that the code to append them to the csv file was our own. For the AI portion, there were a couple of resources. The first was the scikit website so that we could learn how to use the sklearn package to train and test the machine learning aspect of our code. The second resource for this aspect of our code was a YouTube video explaining how confusion matrices work so that we could analyze the effectiveness of the machine learning model. The next part of our code goes into the user interface portion and to do this we used our own code for the data frames and for handling errors. We had to watch a different video about using the Decision Tree Model so that we knew how to actually incorporate the machine learning aspect into the user interface portion of the code. We also included a map of Florida as a reference for the particular city(s) that the user inputs. To complete the map, we used ArcGIS and we had to watch a video about how to use it to find the latitude and longitude of said inputted cities. Furthering that, we had to learn about the folium package from the website that supports that program and we used GitHub source code to learn about how to add markers to the map and for the visualization of the map. The final portion of our code employs the use of graphs to aid in understanding the data overall and the trends that are associated with that data. To complete these, we referenced the textbook chapter that goes over the graphs and visuals and we looked up Jupyter Notebooks specifics to learn about the magic function that makes the graphs interactive for the user.

Methodology

In the previous section of this report, we briefly went over all of the different sections and aspects that our code contains, but here we will go through the processes of getting to this idea and these goals along with going into how we built each cell. The original plan we wrote down was similar to our final but far less thought out to say the least. Originally we were planning on having the user input the different weather conditions and have the machine learning predict whether there was likely to be a natural disaster; this was a flawed idea because this program would essentially need the natural disaster to be currently happening to the user. Furthermore, we planned on it being a worldwide application, which eventually became a national level, which we brought down again to just a state level. After we wrote down this rough idea for out project, we found data sources that we would be able to use; one source for the weather conditions for each day of the year for over 50 years and one for a record of all the natural disasters and which cities they occurred in— both datasets were Florida specific. We also originally had all of our datasets entirely separate thinking it might be easier to reference them, but this was not at all the case.

The next big step in our project, after we had a plan and we had our data, was to work on figuring out how to use the datasets. The first, very time consuming step, was to add the natural disasters data to a different csv file because it was organized differently from our csv files related to the weather attributes of the different cities. We ended up having to add yes/no values to each day in each year that was contained in our data, which made it easier to use in the long run. After this we had to try and call the data into our program and this was the step we pseudo gave up on. We could not figure out how to merge all of the csv files we had into one working dataset that contained everything we needed. Some of the things we tried to do were to make dataframes and to truncate the data, which would not work because whenever we tried to randomize it, there

were never any instances of natural disasters, since they are not everyday occurrences. We also tried to consider the idea of thinking about all of the data as a list and each table associated with a different city was another list and each column was yet again another list, but this was a very roundabout method that did not make sense.

At this point, we had decided to switch our project idea to detecting fake news; it still contained machine learning, which was something we had wanted to incorporate into any project idea we had. This program was working well and it was fully developed, but it was simple and we quickly came to a roadblock in which there was nothing else we could add to the code. We also saw examples of many other projects that did the same thing and we did not feel accomplished by completing the fake news detector, so we reverted back to our original idea and this meant we had to think of other alternatives to solving the problems we had faced when dealing with the data.

Instead of using the many csv files we currently had downloaded for the weather data, we found a different source that was more comprehensive and all we had to do was merge the Florida cities with their respective data, which proved to be a more manageable task for us to complete. Finally, we merged the weather data that contained the cities along with the one other csv file we had left relating to when natural disasters had occurred; which was a somewhat similar process as when the cities were merged. Now that the data was combined, we used the Pandas website to figure out how we should use the code to read the data ("Pandas.read_csv"). Once this was coded, there were still problems with reading the data so we entered the error into ChatGPT and it explained that our data may be incomplete, so we had to fill all the empty cells with 'NO' in the yes/no column of the csv file. There was another problem with the data

averages because they were not making sense; we realized that the dataset we had downloaded filled the empty cells with -99.99. To fix this, we just replaced all of the -99.99 values with NaN:

```
replace_values = {-99.9: np.nan, -99.99: np.nan}
```

(GeeksforGeeks)

The next step in our code was to incorporate the machine learning aspect, but it was difficult to decide which model to use— we had a very trial and error filled process at this point. We used the randomforestclassifier and it gave a very low accuracy, which was disappointing, so we tried the naïve bayes classifier, which did not even work with our dataset. We tried to change the Yes/No column of our data to 1's and 0's instead, but this classifier continued to not work. Finally, we tried the Decision Tree Classifier and it not only worked, but it also had a very acceptable accuracy. We then utilized this classifier to train and test the data; we tested it with 25% of the data ("How to Implement Decision Trees in Python (Train, Test, Evaluate, Explain)"). The last part of the machine learning that we needed to implement was the confusion matrix and the accuracy. Since our data does not have a large amount of "Yes," the matrix was rather limited, but there was not more data to add to it; we then added to the code so that the matrix and the accuracy could be visualized by the user (Dennis, "Confusion Matrix Visualization").

```
# Predictions and evaluation
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
matrix = confusion_matrix(y_test, y_pred)
```

Another aspect we wanted to have in our project was a user-interface, because that makes it more interactive and makes it easier to comprehend the datasets that we had gathered. We tried

to make widgets and we also tried to connect the machine learning to a website using Flask and Ngrok, but all of these attempts were unsuccessful (required HTML) so we ended up just integrating the user interface directly into Jupyter Notebook. This lacks some visual appeal, but it is something we could expand on in the future when we have more time to experiment. We integrated the data, added comparisons of two cities, and included the predictive aspect from the machine learning part of the code. We added the comparison of the two cities after the fact to add an additional feature for the user. Going into the user input, we had to do some error handling with the name of the city, so when the user inputs a city it should be capitalized.

We have two major elements left, and both are visuals that rely on the matplotlib package. The first is the map of Florida that includes markers of the cities that the user inputs. The difficulty with the map is that, for the markers, not only the city name is needed, but also the latitudes and longitudes. We initially tried to establish an API; we tried this using Google Maps, Bing, and several other platforms but every time we tried to use the API key we were denied and the code was inconclusive. Instead, we found a way to use an ArcGis extension and it was very useful and essentially found the city locations (longitude and latitude) without needing an API key. The map visualization is produced using the folium package and the markers, or popups, show the cities that the user inputted, as well as the risk for a natural disaster in those cities on a user inputted date.

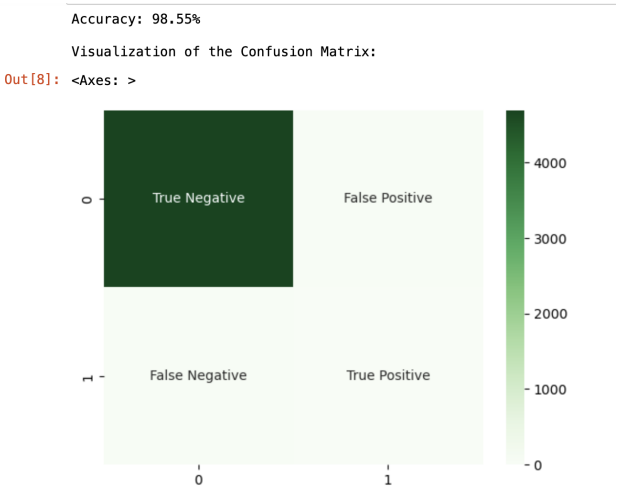
The last part of our code is simply data analysis and it produces the graphs for the different weather attributes for the days of each month. The data presented is truncated because with all the data points present, which was the original goal, it looked just like a blue square because there were too many points to meaningfully visualize. The graphs are also presented in a 3D format so that multiple aspects can be compared; for example, the user can view the

maximum temperatures compared with each day of the months, or just the months themselves. At first, we had done months and years, but this data did not show trends as clearly due to the fact that we have many years included in our data set which can overwhelm the visual. This part required us to consult the textbook so that we could ensure that the graphs were built properly (Kong, et al., “2D Plotting — Python Numerical Methods”).

Results

The results of our work are somewhat subjective, but we were able to complete all the goals we had set out for our project. The main point of success for us was being able to merge all of our data and turn it into a csv file that was readily able to be called and used within our code, but that is not something that is easily visualized. Instead we’ll focus on the user interface and the accuracy of our machine learning model in this section of our report.

As previously mentioned in our methodology, we tried multiple machine learning models and only one of them had accuracy that was acceptable. We cannot really explain the why behind these models because that is beyond the scope of our programming knowledge at this point in time, but we were able to visualize a confusion matrix that displays the accuracy:



The next part of our results that can be demonstrated visually is with the user inputs. We have are single city user input which will predict if there is likely to be a natural disaster on a specific date in the future and there is also a way to compare two cities, along with outputting the prediction based on the machine learning model (Decision Tree Classifier). Here is one example of the city comparison:

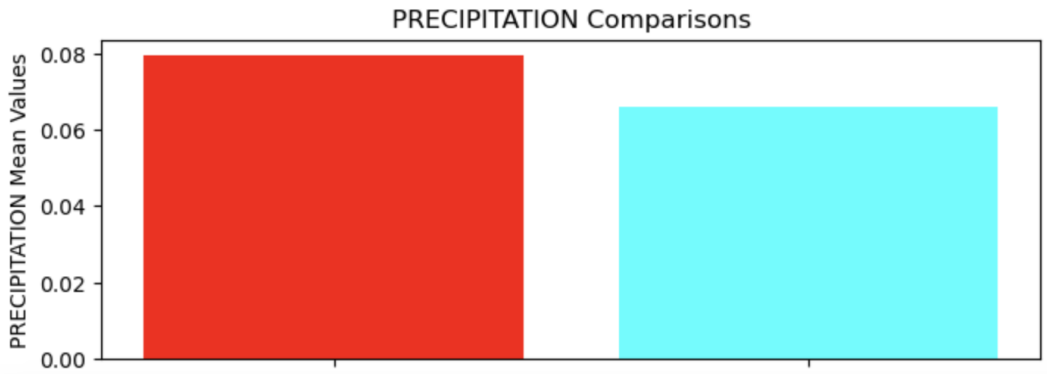
```
Enter the first Florida City: Gainesville
Do you want to enter a second Florida City? (YES/NO): YES
Enter the second Florida City: Jacksonville
Enter Year: 2030
Enter Day: 1
Enter Month: 1

For Gainesville, it is not likely for there to be a natural disaster.


For Jacksonville, it is not likely for there to be a natural disaster.

The mean precipitation for City 1: 0.0796969696969697
The mean precipitation for City 2: 0.0661111111111111
Difference in Mean precipitation: 0.01358585858585859

The mean mean temp for City 1: 71.28848484848484
The mean mean temp for City 2: 72.445
Difference in Mean mean temp: 1.1565151515151513
```



Overall, our results were successful and matched our end goals, despite the large amount of trial and error described in the methodology.

Conclusion and Future Work

Our code accomplishes all of the goals we had for it, but given more time and experience it has the potential for many other functionalities. We were able to merge all of the data to make it usable, which was a very difficult process that had us nearly on the verge of giving up on this project idea, and we were able to use that data to make a user interface, develop machine learning, and for data analysis. We enjoyed being able to complete our goals and weather data was a great way to experiment with new packages and functions that we have not used before. We would like to expand the machine learning aspect and learn more about how that works—like how we chose the Decision Tree Classifier, but we do not really know why that model has the most accuracy.

In order to further our code in the future, there are several features we could add to and create. Right now the user has the ability to input up to two different cities in Florida and one way to expand the berth of this aspect would be to add the number of cities the user can input and maybe to work to find data and expand into other states. It would also be fascinating if we could find more data about the weather for the cities we already have so that we could merge that and include more analysis; one example would be adding barometric pressure as a column in our data. One thing we would be especially interested in doing in the future would be adding a visual user interface, such as a website, so that it would be more engaging and familiar to use to a wider audience. Overall, we had a great experience with this code and we learned more about the expansive uses that Python has to offer for programming.

Works Cited

"Climate Data Access Tools." *Florida Climate Center*, FSU, climatecenter.fsu.edu/climate-data-access-tools/downloadable-data. Accessed 10 Dec. 2023.

CodeBasics. "Machine Learning Tutorial Python - 9 Decision Tree." *YouTube*, YouTube, 17 Nov. 2018, www.youtube.com/watch?v=PHxYNGo8NcI.

"Florida, USA ." *Lat Long Finder*, www.latlong.net/place/florida-usa-15262.html#:~:text=Latitude%20and%20longitude%20coordinates%20are,and%20the%20Gulf%20of%20Mexico. Accessed 10 Dec. 2023.

"Getting Started." *Getting Started - Folium 0.1.Dev1+g0f4d57f Documentation*, Folium, python-visualization.github.io/folium/latest/getting_started.html. Accessed 10 Dec. 2023.

Kong, Qingkai, et al. "2D Plotting — Python Numerical Methods." *Python Programming and Numerical Methods*, Elsevier, pythonnumericalmethods.berkeley.edu/notebooks/chapter12.01-2D-Plotting.html. Accessed 10 Dec. 2023.

"Pandas.Dataframe.Fillna." *Pandas.DataFrame.Fillna - Pandas 2.1.4 Documentation*, pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html. Accessed 10 Dec. 2023.

"Pandas.DataFrame.Merge." *Pandas.DataFrame.Merge - Pandas 2.1.4 Documentation*, 2023, pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html.

"PANDAS.READ_CSV." *Pandas.Read_csv - Pandas 2.1.4 Documentation*, pandas.pydata.org/docs/reference/api/pandas.read_csv.html. Accessed 10 Dec. 2023.

shardul_singh_tomar. "Replace Nan Values with Zeros in Pandas DataFrame." *GeeksforGeeks*, GeeksforGeeks, 9 May 2023, www.geeksforgeeks.org/replace-nan-values-with-zeros-in-pandas-dataframe/.

"Sklearn.Preprocessing.LabelEncoder." *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed 10 Dec. 2023.

Softdemic. "How to Plot Latitude and Longitude on the Map Using Python.#python #lambda #coding #geopy #folium." *YouTube*, YouTube, 4 Sept. 2021, www.youtube.com/watch?v=g4rY-dGkBw8.

T, Dennis. "Confusion Matrix Visualization." *Medium*, Medium, 25 July 2019, medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea.

Turp, Misra. "How to Implement Decision Trees in Python (Train, Test, Evaluate, Explain)." *YouTube*, YouTube, 11 June 2021, www.youtube.com/watch?v=wxS5P7yDHRA.