CPSC 4310
Project Report
Written By: Dawson Meyer and Eric Den Haan

Implemented Component

We chose to adapt and implement the strategy described in *K-Nearest Temperature Trends: A Method for Weather Temperature Data Imputation* (Kiani and Saleem, 2017). This paper gives a method for predicting the temperature measured at a weather station in the case that there is missing data for a certain date. Other techniques currently in use involve gathering data from nearby weather stations to fill in missing temperatures, but this technique allows a single station to be independent.

The algorithm proposed by the researchers begins by locating a missing value. A date range is also given as input to the algorithm. For example, we may want to consider 3 days before and after the target date. That date range is located for the other years in the dataset and temperature "trends" are computed. Assuming that the other years in the dataset were not missing values for the same date, the trend is computed by determining the difference in temperature between the missing date in the other years and the other dates in the date range. For example, if the missing value was April 7, 2017, then the difference between April 7, 2016 and April 6 and 8, 2016 would be calculated, and so on for the rest of the range given (i.e. 3 days before and after April 7). These distances would then be computed for each year in the dataset (other than 2017) and inserted into a table. Once this process is completed, the average temperature difference is calculated for each year. The years which have the lowest average temperature difference from the target date over the date range are considered to be the most similar. The arithmetic mean of the temperatures in these similar years on the target date is taken, and this is given as the estimated value for the temperature on the missing date.

Our Implementation

We used a dataset from the American NOAA (National Oceanic and Atmospheric Administration), which allowed us to gather daily maximum temperature data for the Great Falls, Montana weather station from 1990-2004. 15 years of data is the same amount used by the researchers. At first, we used a much more comprehensive dataset containing various temperatures, humidity, and other information, but its size (25 mb .csv file) meant that the NOAA could only provide 10 years worth of data. Since we only needed daily temperature maximums, we settled on the simpler dataset that would be able to give a wider range of data. Because the dataset had no missing values, we were able to have a solid basis on which to evaluate our algorithm for accuracy. We added an extra month to each end of the dataset to allow imputation for January 1, 1990 and December 31, 2004 as the algorithm needs a certain number of dates before and after the target date (provided by the user) to be available in the dataset. We used Python as our programming language of choice due to the large number of built in libraries for handling .csv files as well as the ability to write a working prototype very quickly with very little overhead needed for language constructs.

The first part of our implementation involves parsing the data from the dataset and inserting it into a data structure that will be easily accessible by the rest of the algorithm. The `parseData()` function handles this part of the strategy. The data structure we chose to is a `dictionary` where the key is the date (as a `datetime` object) and the value is the temperature for that date. In this way, we would be able to quickly look up temperatures on a given day in our dataset and use the built-in methods for date arithmetic while performing calculations.

The second part of our implementation deals with the actual estimation of the missing data. There were a few things that had to be dealt with before we could actually infer the value of the missing date. Mainly, there was the problem of dealing with leap years. The problem with the proposed method is that the researchers don't say how they dealt with leap years in their experiment. What we decided to do was compare February 29th from a leap year to February 28th from non leap years so that we can still have somewhat accurate results for that day.

The `findAvg()` function is where we calculate the temperature imputation from the parsed data. The first step in this function is to find the dates that match the target date for the other years in the data set. They get put into the `listOfDates` list in order to have the list of dates that we can work with from every year. This list is sorted so that we can look through the data sequentially. Next, we take the user input for how many days before and after to consider, and we create a list that is populated by the temperatures from those days from each year. The temperatures of the range of dates for the original year is then compared to the same date for each year, for example, August 12 for the year with the missing date is compared to all of the other August 12ths in the data set. The absolute value of the difference between these values is recorded in the `diffList`. The differences are then averaged for each year in order to see what years are the most similar to the original year for that range of dates. We define "similar" here as the lowest values in `diffList`. The three most similar years are chosen as the values to base the imputation on. Once these years are chosen, we find the month and day that are the same as the missing date from those years, and we average the temperature for those three days which ends up being our imputation value.

Evaluation Strategy

The researchers use various techniques for evaluation including Root Mean Square Error and comparison to other implementations of temperature imputation. Because we had a large range of data and no missing values in our dataset, we decided to take advantage of this and show the difference between the estimated temperature and the actual value for all of the dates in the dataset. In *single_item.py*, the algorithm is evaluated simply by providing this difference for a single temperature to a user. In *full_analysis.py*, the `runAnalysis()` function first calls the `findAvg()` function on every item in the dataset, then writes the date, actual and estimated temperatures, and the difference between them. At the end of the log file, the average difference is calculated, and this is used to determine the accuracy of the algorithm.

Conclusions

The first main challenge we faced with this project was tracking down a dataset that would work well for our purposes. We can see how gathering reliable data would be one of the biggest challenges in any data mining effort. Our evaluation method did allow us to find areas where our algorithm was not performing as expected and could be improved. We brought the average temperature difference between estimated and actuals from approximately 12 to 7 degrees fahrenheit, which is in line with the results found by the authors. Changing the date range for analysis did not produce any accuracy improvements. An interesting hypothesis to test would be to see whether the algorithm performs more accurately in a place with a smaller temperature range (such as Arizona) than it did in Great Falls, which can have a wide variance in temperature from summer to winter. Overall, it was interesting and rewarding to implement this intriguing data mining strategy.