Weather Predictor

Software Design Document

Simardeep Khinda

03/16/2023

CS 225, Spring 2023

Embry-Riddle Aeronautical University

Daytona Beach Campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

**Introduction:**

This Software Design Document (SDD) has been constructed to describe the responsibility and the design of the software. A weather forecaster is a prediction of the future weather statistics. For this design, the generated data will be provided by the user via an input file. The user will provide the software with data from 20 prior days. Using that data, the software will use 2 algorithms to predict the next 10 days’ data. Furthermore, the software will use a 3rd method which will have the average of the previous two methods.

The weather forecaster project has been designed as a project for CS 225. It represents the ability to generate forecasted data using predictive modeling techniques. Alongside, the versatility of the project can be applied to many other concepts.

**Problem Description:**

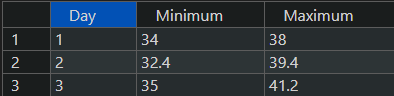
The weather forecaster bases its data and statistics primarily on the original user data. The user will provide the software with 10 days (about 3 weeks) of maximum and minimum temperature or more. 10 days is the minimum, so the software has enough data to make an accurate prediction. The more data that is provided, the more accurate the predictions will be. The predicted temperature will be calculated through the three methods as shown below in table 1.

Table 1

|  |  |
| --- | --- |
| **Weather Forecaster** | **Forecaster Behavior** |
| LinearRegression | This class takes the input of user data and creates a slope and intercept to put together an equation. Using that equation, the forecasted data can be predicted. |
| PolynomialRegression | This class takes in the input of user data and finds the relationship between the day and the value of the temperature. The algorithm finds a non-linear regression line which can be used to identify patterns that may be more complex for linear regression. |
| Average of linear regression and polynomial regression. | This class takes the result of LinearRegression and PolynomialRegressionand finds the average between the two. This allows for a mean between the two algorithms above. |

The user inputted a CSV file called “input.csv” containing maximum and minimum temperature for the previous 20 days. Here is an example below (Figure 1):

**Figure 1**



Using the provided information, the software reads and processed it to provide predicted temperatures for minimum and maximum for each day. Specifically, the software processed the data from days 1-20. After it trained the models and used them to predict, it calculated the predicted minimum and maximum temperatures from days 21-30.

With the three provided methods of calculating and predicting the future weather, the user had the opportunity to see how various algorithms analyzed the data and resulted in different outcomes. The data may have become more inaccurate as the days increased but with multiple methods, it followed the trend(s) while keeping it simple for the user to view.

The calculated data was output by the software to three different files, each for the different methods of predicting. The file names were as followed: “LRPrediction.csv,” “PRPredicition.csv,” “AveragePrediciton.csv.” They had the same format as figure 1.

This software demonstrates the power and ability of prediction using algorithms. Applying them to programming allows for the user to be able to predict a set of numbers based on historical data. Such a program can be modified for one’s use to predict almost any data type.

**Problem Solution:**

The temperature is predicted through two methods, along with an average of both data. The first method of prediction is **linear regression**, and the second method is **polynomial regression**. The UML (Unified Modeling Language) for the software is attached below:

Diagram

Description automatically generated

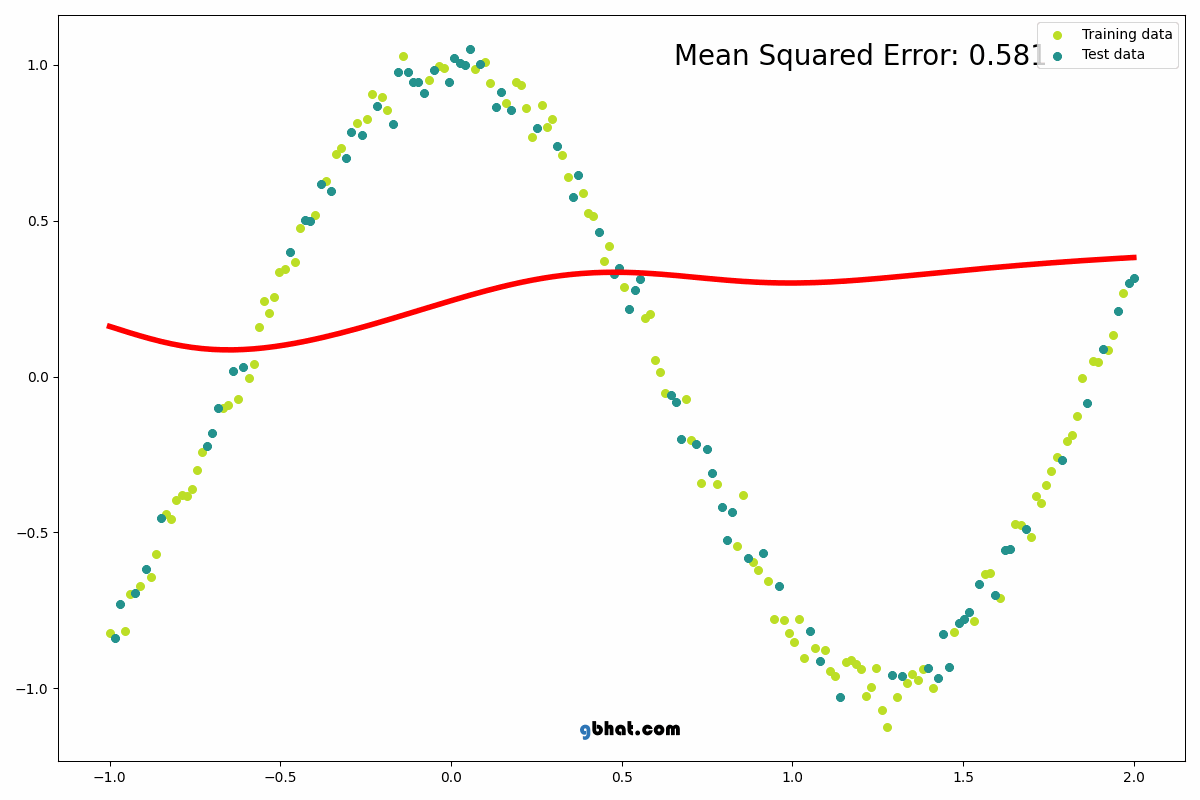
The **linear regression** method is calculated using the equation of a line which requires the variables, **m** (slope) and **b** (y-intercept)to make a prediction for the weather. Below is the equation to find the slope (**m**)

**L** is the number of given days (10 in this case). **x** is the day and **y** is the temperature for the respective **x**. For instance, in **(x, y)** format, (5, 37.5) can be interpreted as “The temperature for day **5** is **37.5** degrees.”

After the slope has been calculated, the intercept is needed. Below is the formula to determine the intercept (**b**)**:**

The slope and intercept can be put together to form the equation of a line. For each coordinate, the equation can be used to predict the temperature and other data types. Below is the equation:

**Polynomial regression** takes in the day as the **x** variable and the temperature as the **y** variable. It is used to capture non-linear relationships that may not be identified by linear regression. Unlike the linear regression where a linear line is fit into the graph to identify a trend, polynomial regression fits in a curve which allows for more accurate predictions towards trends. Just like linear regression, outliers may cause the predictions to be inaccurate. Future implementations can be made for the data to be analyzed with moving averages to smoothen out the data before it is passed on to polynomial regression to eliminate outliers impacting the predictions.

Below is a visualization of what the line would look like for a polynomial regression:

To implement this to code, Apache’s Commons Maths library is used. It is used to set the weighted observed points. These are points that gain more weight as the days increase as the software shall look at more recent data points for more accurate findings. Using the observed points, the software fits a curved line to determine the trend and linearity.

Below is a description of each of the classes:

Controller Class:

This class controls all the aspects of the software. It calls upon the other classes to predict the temperature and other data types. The class is responsible for the following:

1. Creating an instance of the FileManager class.
2. Creating an instance of GenericPredictor for each set of data.
3. Call the FileManager class to return the inputs of the CSV file.
4. Call LinearRegression and PolynomialRegression classes to predict the data and output to file using FileManager.
5. Calculate an average between the predicted two types of data and write to file.

GenericPredictor Class:

This class is only used as a base for LinearRegression and PolynomialRegression. It will not be called upon; it is just a template for inheritance.

1. Receive user input from Controller class and store in an array list of doubles called oldData.
2. Initialize the train() method.
3. Initialize the predict() method.
4. Create a method that can be called by children to only have two decimal points in a double value.

LinearRegression Class:

This class is responsible for linear regression to train the data and predict the future forecast.

1. Class shall be extended from GenericPredictor
2. Class shall define the train() method with the linear regression algorithm.
3. Class shall define the predict() method to predict the next 10 days of data.

PolynomialRegression Class:

This class is responsible for polynomial regression to train the data and predict the future forecast.

1. Class shall be extended from GenericPredictor
2. Class shall define the train() method with the polynomial regression algorithm.
3. Class shall define the predict() method to predict the next 10 days of data.

FileManager Class:

This class is responsible for inputting and outputting to a CSV file.

1. Class shall read from file and pass data where needed.
2. Class shall write to file with the provided data and parameters from software.
3. Class shall read the temperature and be able to read minimum and maximum temperature separately.

**References:**

All sources shall be cited in this section. If the project required no sources, keep this section but leave it blank. Sources might be papers and texts in the general problem domain of the project, code snippets, libraries incorporated in the project, or even algorithmic solutions to specific parts of the project.

* <https://www.mathsisfun.com/data/least-squares-regression.html>
* https://www.analyticsvidhya.com/blog/2021/07/all-you-need-to-know-about-polynomial-regression/
* https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/fitting/WeightedObservedPoints.html
* https://commons.apache.org/proper/commons-math/userguide/fitting.html

**Appendices:**

This is optional, but may include external sources, source code, or other related material.

*[Shall be completed as needed with each deliverable.]*