Outdoor Air Temperature Prediction
Phase Four

CSCE 3602 Fundamentals of Machine Learning

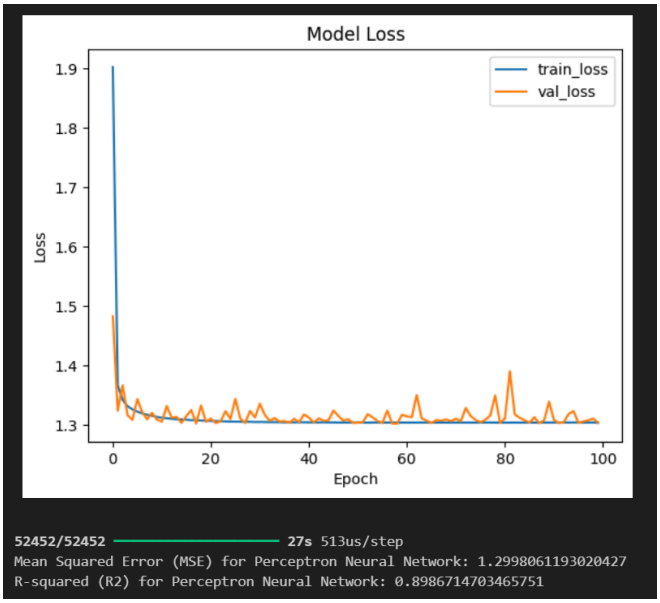
Amr Abdelbaky
Computer Science
The American University In Cairo
Cairo, Egypt
amrkhaled122@aucegypt.edu

Ali Eissa
Computer Science
The American University In Cairo
Cairo, Egypt
alieissa@aucegypt.edu

Before Starting This Milestone :

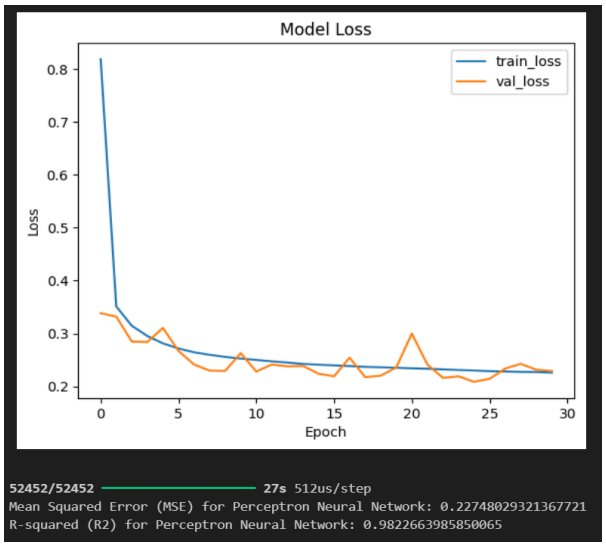
According to the provided feedback from milestone 3 , we are currently experimenting with NN with the provided feedback (linear activation on hidden layers and a modified loss function) we are also implementing the DT model with the customized gain function to take care of the extremely low and extremely high temp value that can cause imbalances in our model training, it multiplied the losses by 3 for such occurrences.

This was the result of the NN model with linear activation and weighted loss function for 100 Epochs:



MSE	R2
1.299806119302	0.89867147034

This was the result of the NN model with RELU activation and weighted loss function for 30 Epochs:



MSE	R2
0.22748029321	0.982266639858

The results still came short in MSE and R2 scores compared to the previously chosen Decision Trees model with : **MSE** = 0.10388761598921503 and **R2=** 0.9919012695654646 ; even after modifying the Loss function and using linear activation , So we are proceeding with The **Decision Trees** model with modifying the gain function to increase the weight of error on the periphery of our dataset due to some imbalances on the dataset to make sure our model captures this part of the Dataset correctly .

Justifications:

Decision Tree Regression models serve as fundamental tools in the realm of supervised machine learning, particularly in addressing regression tasks where the objective is to predict continuous outcomes(such as temperature in our case) . This architectural framework embodies a hierarchical structure rooted in principles of recursive partitioning and feature-based decision-making. They work well for such tasks because they partition the data into subsets using the most predictive features, making them ideal for capturing the complex dynamics of environmental data for example.

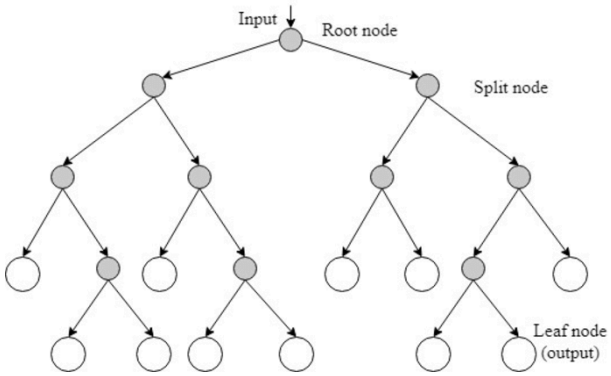


Image source

At the core of the Decision Tree architecture lies its tree structure, characterized by nodes representing feature attributes and edges embodying decision rules based on feature values. Commencing with a root node, the tree undergoes iterative partitioning, culminating in leaf nodes that encapsulate final predictions. These nodes, including root, internal, and leaf nodes, collectively define the topology of the decision tree.

Central to the efficacy of Decision Tree Regression is the criterion employed for node splitting. While various metrics exist, including Gini impurity and entropy for classification tasks, regression endeavors typically hinge upon minimizing the Mean Squared Error (MSE) within each partition. The decision rules established at internal nodes guide this partitioning process, dictating feature thresholds and subsequently segregating data subsets.

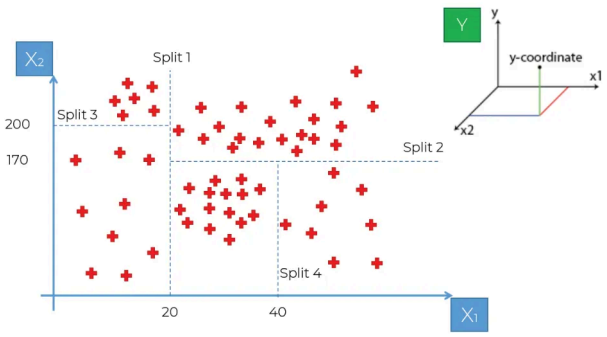


Image Source

The training process of a Decision Tree entails iterative optimization, where the algorithm selectively identifies optimal feature-threshold combinations to minimize the chosen splitting criterion. This iterative refinement is pivotal in shaping the tree's structure, ensuring that subsequent predictions reflect an accurate representation of the training data's underlying patterns.

Prediction with Decision Trees involves traversing the tree from root to leaf based on the feature values of a given instance. At each node, decisions are made according to the established rules until a leaf node is reached, where predictions are generated. Typically, predictions at leaf nodes entail aggregating target values of training instances associated with that particular node.

Furthermore, the architecture of Decision Tree Regression models is governed by hyperparameters, which regulate the complexity of the tree and mitigate overfitting or underfitting tendencies. Parameters such as maximum depth, minimum samples per leaf, and minimum samples per split exert influence over the tree's growth, thereby shaping its predictive capabilities.

Our Finalized Model Design and Architecture

1. Implementing Custom Weighting in the Loss Function

We are planning to address the challenge of data imbalances at the dataset's periphery by implementing a custom loss function in the training phase of our Decision Tree model. The intention is to dynamically adjust the weights for training instances that fall into extremely cold and hot temperature categories. This adjustment will be based on the deviation of these temperature values from the median. By assigning higher weights to greater deviations, we aim to enhance the model's focus on these critical areas. This strategic modification is expected to improve the model's accuracy in predicting temperatures at these extremes, ensuring that these underrepresented data points do not diminish in influence during model training.

2. Validation Strategy

To evaluate the effectiveness of our Decision Tree model, particularly with the new weighted loss function, we plan to implement a k-fold cross-validation strategy. This approach will allow us to comprehensively assess the model's performance and ensure it generalizes well across different segments of the data, especially in accurately capturing temperature extremes impacted by our loss function adjustments.

3. Hyperparameters Tuning

As we are going to build the model from scratch , in the meantime we are going to perform a grid search to find the best hyper parameters using sci-kit library in python to save time and experiment with different combinations of hyperparameters in order to use the best in our model that will be built from scratch, these are our current hyperparameters with justifications for them.

- **Max Depth**
  We set the maximum depth to 20 to allow for complex rule learning. This depth helps in capturing detailed patterns but poses a risk of overfitting, which we monitor through cross-validation to ensure the model generalizes well, we might consider reducing this max depth .
- **Criterion**
  We currently continue to use squared_error as the criterion, focusing on minimizing the mean squared error within each node, which is ideal for our regression tasks to improve prediction accuracy.
- **Splitter**
  We currently selected the best splitter to ensure optimal split decisions at each node, aiming for the most significant reduction in variance and enhancing model performance. This method is computationally more intensive than others like random, but it provides better split decisions that can improve the model's overall predictive performance.
- **Min Samples Split**
  Originally set to 2, we might need to increase this number to reduce overfitting risk. A higher threshold ensures that splits are based on more substantial grounds, improving model robustness.
- **Min Samples Leaf**
  We might need to increase from 1 to prevent overly granular rules and overfitting, making the model more general and robust by avoiding leaves with very few samples.

These changes to the hyperparameters are strategically chosen to balance the complexity and generalizability of the model, enhancing its performance and reliability.

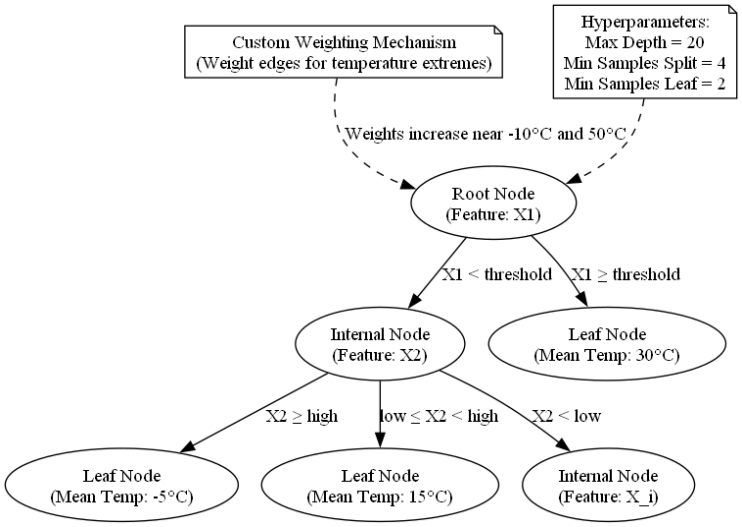
4. Pruning Strategies

To optimize the Decision Tree model further, We are considering the implementation of pruning algorithms. Pruning helps in reducing the complexity of the final model by removing sections of the tree that provide little power in predicting the target variable. This process not only helps in combating overfitting but also in improving the overall efficiency of the model. Two common pruning techniques are:

- **Cost Complexity Pruning (CCP): Also known as weakest link pruning:**
  This method is applicable to both classifiers and regressors. For regression trees, the complexity parameter, often denoted as alpha, balances the tree's depth against the fit to the training data. Nodes are pruned if they do not contribute to a decrease in the overall error metric (like MSE) greater than the cost defined by alpha. This is part of the CART (Classification and Regression Trees) algorithm and is indeed used to prevent overfitting by simplifying the model.
- **Reduced Error Pruning:** This is more commonly mentioned in the context of classification trees. For regression trees, pruning cannot be based on the most popular class since the output is continuous. Instead, it involves collapsing nodes based on whether doing so reduces the prediction error on a validation set or does not increase it beyond a certain tolerance. Nodes or subtrees are replaced with leaves. The value at the leaf may be the mean outcome of the samples that fall into that region, and the change is kept if it does not significantly increase the error.

Implementing these pruning techniques during the post-building phase of the tree construction will be explored to determine their impact on the model's performance and reliability.

This is an Imaginary look of how a subset of our model will look like at the end of its implementation :



Utility Application:

Design

The following Sequence provides the utility application design on how the user is going to interact with our model in a basic Client/Server design. The backend will host our prediction model and the user will interface through a UI. The first step is to enter the Meteorological Features and the region in their raw from the client interface of the application , where the end user only deals with this interface , the raw data consists of the following 23 features:

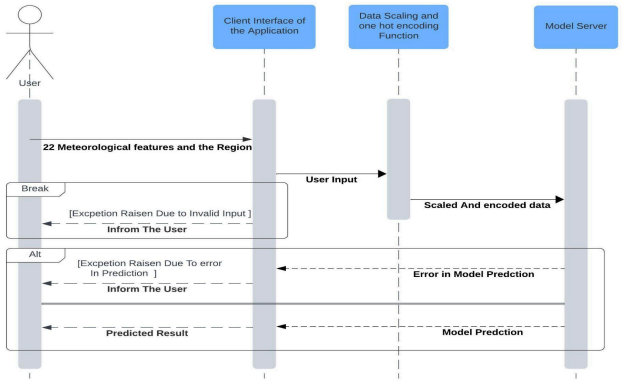
1. Hour.
2. Amount of precipitation in millimeters (last hour).
3. Atmospheric pressure at station level (mb).
4. Maximum air pressure for the last hour in hPa to tenths.
5. Minimum air pressure for the last hour in hPa to tenths.
6. Solar radiation KJ/m2.
7. Air temperature (instant) in celsius degrees.
8. Dew point temperature (instant) in celsius degrees.
9. Maximum temperature for the last hour in celsius degrees.
10. Minimum temperature for the last hour in celsius degrees.
11. Maximum relative humidity temperature for the last hour in %.
12. Minimum relative humidity temperature for the last hour in %.
13. Relative humidity in % (instant).
14. Wind direction in radius degrees (0-360).
15. Wind gust in meters per second.
16. Wind speed in meters per second.
17. Latitude.
18. Longitude.
19. Height.
20. Year.
21. Month.
22. Day.
23. The state of the user in the north of Brazil (one of seven states which are) which is a choice of seven states , which are as follows :
    1. AC: Acre.
    2. AM: Amazonas.
    3. AP: Amapá.
    4. PA: Pará.
    5. RO: Rondônia.
    6. RR: Roraima.
    7. TO: Tocantins.

The Interface is then to send these raw data to a function in the application to apply the required scaling and one hot encoding to the given data in order to provide it as an input to our model server , the model server is then to predict the Temperature (instant) in Degrees celsius and Return it back to the Client Interface in order for the User to receive the output .

The Sequence diagram also addresses the possibility of having an error in the input (not entering a feature or entering an invalid value for some features) , another error possibility is that after parsing the input from the user and performing all data preparations, the model server itself might provide an error because it is not 100% fault tolerant (nothing is) , in this two cases of errors, an exception is raisin and the user is informed that an error occurred during this instance of usage .

We might also consider making it easier for the user by substituting some of this data from a real-time weather API like: https://openweathermap.org/current however, our implementation of this will depend on the timeframe for the next milestone.

The Sequence Diagram :



Architecture :

Client (Webapp UI): The user interface where clients can input data, send requests for predictions, and receive results. It's designed for ease of use and doesn't handle any of the predictive logic.

Server (Prediction Model and Backend Logic):

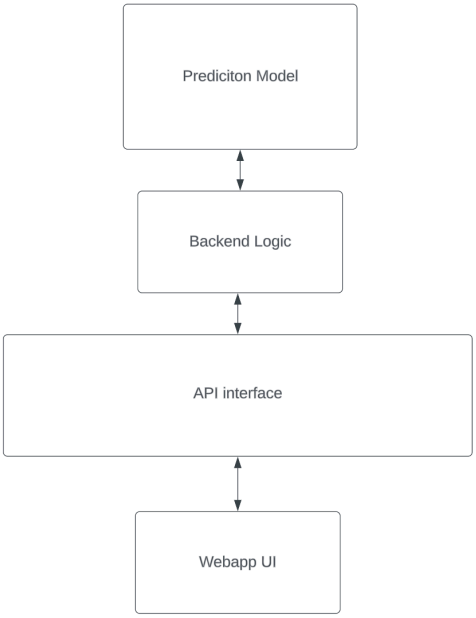
- **Prediction Model**: Resides on the server and is the core of your application. It receives processed data, performs the prediction, and sends back the results.
- **Backend Logic**: The intermediary layer that handles communication between the prediction model and the API. It processes incoming requests, executes necessary data transformations, invokes the prediction model, and formats the predictions to send back to the client.

API Interface: Acts as a conduit for requests and responses. It defines the endpoints through which the client communicates with the server. This interface is crucial for defining how the client's requests for predictions are received and how the results are returned.

This client/server architecture ensures a separation of concerns, allowing the server to handle complex operations and the client to focus on providing a responsive and intuitive user interface.

The model's maintenance, updates, and scalability are managed server-side, abstracting these complexities from the client.

The following is a Simple representation of this architecture.



References:

Lin, TC. (2023, January 10). *Decision tree regression: Machine Learning*. Medium. https://medium.com/@chuntcdj/decision-tree-regression-machine-learning-337abfc2ba4e

Prasad, A. (2024, February 7). *Regression trees: Decision tree for regression: Machine Learning*. Medium. https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047

Wikimedia Foundation. (2024, February 4). *Decision tree pruning*. Wikipedia. https://en.wikipedia.org/wiki/Decision_tree_pruning#:~:text=Reduced%20error%20pruning,-One%20of%20the&text=Starting%20at%20the%20leaves%2C%20each,advantage%20of%20simplicity%20and%20speed.