PREDICTIVE ANALYSIS OF WEATHER CONDITIONS USING GRADIENT BOOSTING

WEIDEMAN, JOHN-PIERRE

Table of Contents

Introduction

We conduct a predictive analysis of weather conditions for the city of Szeged, Hungary, using a decade's worth of hourly meteorological data from 2006 to 2016. The primary objective is to develop and optimize two separate Gradient Boosting models to predict temperature and apparent temperature, respectively. The dataset includes various numeric and factor variables.

The analysis begins with data cleaning to ensure the dataset is properly formatted, free of anomalies, and ready for modeling. Special attention is given to imputing missing data using the proper techniques. We also handle categorical variables to ensure their suitability for our modeling purposes.

An exploratory analysis is conducted to explore the relationships between variables to identify potential predictors and refine the dataset for prediction.

The modeling phase involves the optimization of hyperparameters, including the number of trees, maximum tree depth, and shrinkage rate, using grid search. We train our model and explore the gains in model performance resulting from cross-validation. Separate models for temperature and apparent temperature are developed. The results highlight the most influential variables and their connections to weather patterns in Szeged.

1) Exploratory Analysis

Data cleaning and exploratory analysis is done in Data_cleaning.r and Exploratory_analysis.r respectively.

1.1) Correcting the data format

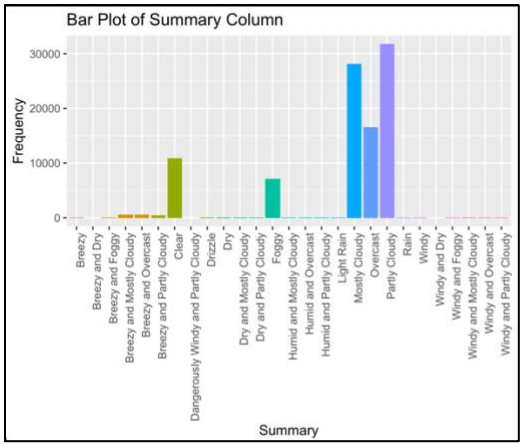
Firstly, to conduct exploratory analysis, we must clean the data and format it in a way that allows us to do the analysis.

- We correct the format of the columns. We make sure that the columns are converted to factors, numerical values, integers etc. according to the data type.

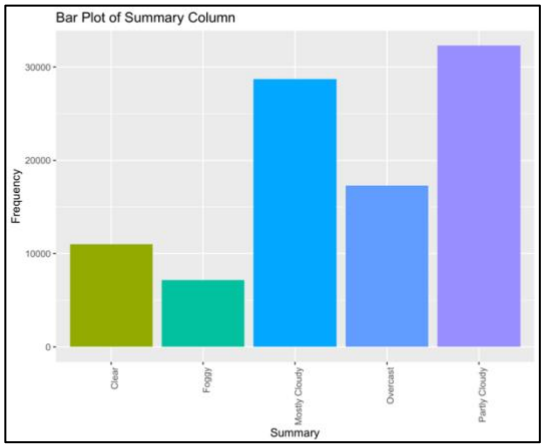
1.2) Cleaning each column of the data, conducting exploratory analysis and dealing with missing/anomalous data

Each column is cleaned separately as follows:

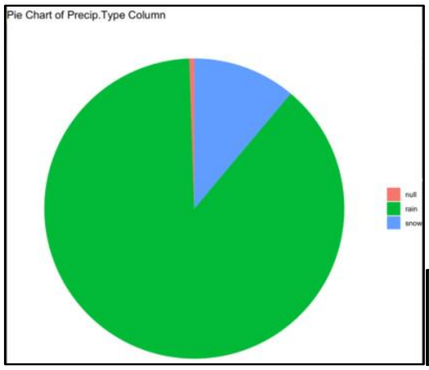
- Summary: Summary is a column with 27 factor levels.

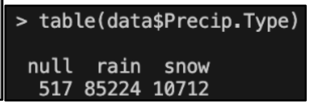


However, some levels contain very few entries, so for improved data analysis, we modify the column by aggregating the levels:



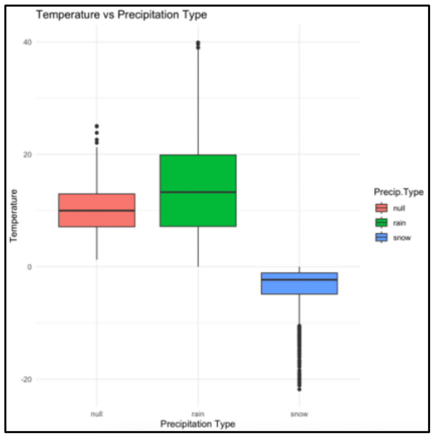
- Precipitation Type: We see that there are 517 values in the Precipitation Type column where there's no precipitation. This is only 0.5360124% of the values. This is a very negligible amount of data points with no precipitation (see pie chart).



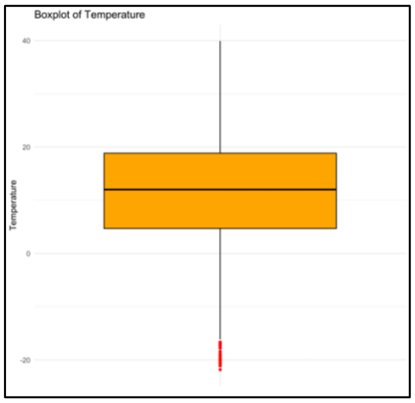


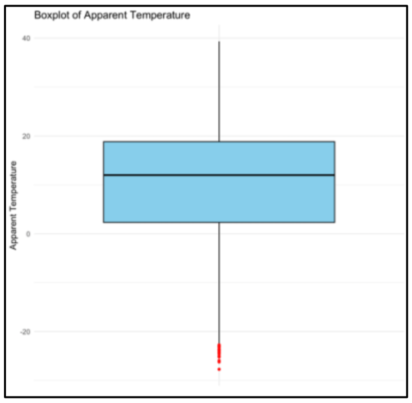
```
> table(data$Precip.Type)

 null  rain  snow
  517 85224 10712
```

In the box plot below, we see that the null values (values with no precipitation) and its temperature distribution is closer to that of rain than snow. Therefore, merging null with rain is a reasonable approach which allows us to simplify the column into two categories (rain and snow).

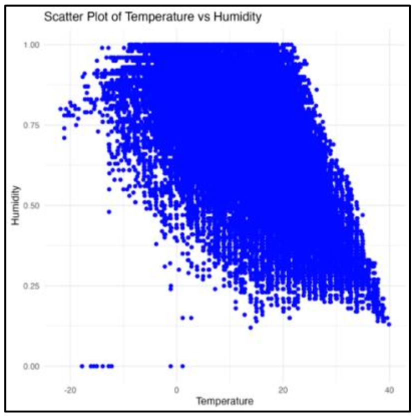


- Temperature and Apparent Temperature: Temperature and Apparent Temperature are our dependent variables. From the box plots we see there is a few outlier values with very cold temperatures. However, these values are realistic values and there is no reason to believe it is mistaken inputs. There are also no missing values. We thus leave these columns as they are.

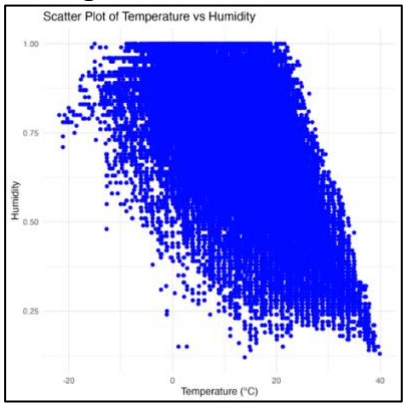




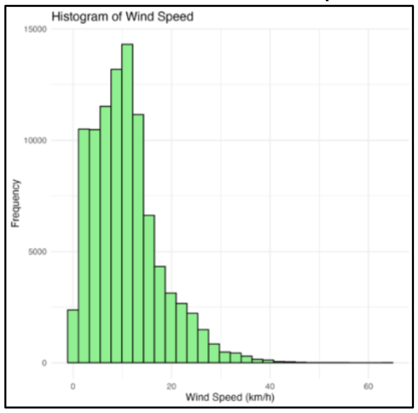
- Humidity: Looking at the scatter plot below, we see that there are some 0 values which seems to be anomalous entries. It is not possible to have 0% humidity in reality.



We thus treat the 0 values as missing values. And since these values are missing at random (MAR) we use the method of Multiple Imputation by Chained Equation to impute the missing values. We impute the values using the "mice" package. We use this method instead of the EM-algorithm because the em.norm() function assumes the data follows a multivariate normal distribution. Humidity, however, is constrained to the range [0, 1], making it unsuitable for direct imputation using a normal distribution.

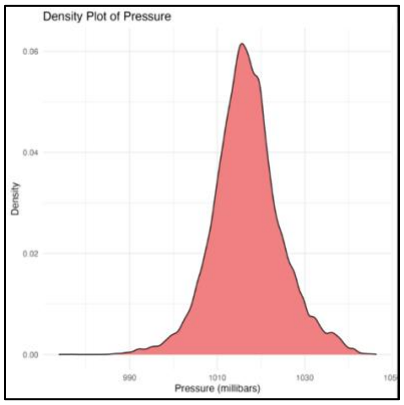


- Wind Speed: There are no missing values. There are 1297 values with 0 km/h wind speed. But unlike humidity, it is possible to have 0 values for wind speed, so we don't consider these values anomalous. We don't need to clean this wind speed column.



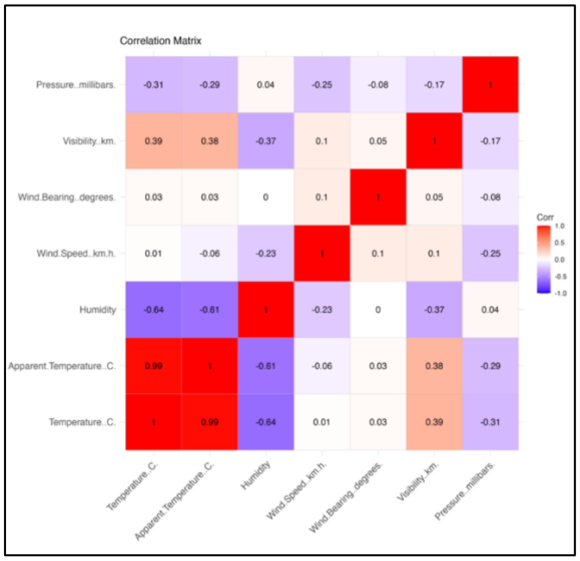
- Wind Bearing: There are no missing values and no anomalous values, there's no need for data cleaning on this column
- Visibility: The visibility column has no missing or anomalous values.

- Cloud Cover: The cloud cover column only contains 0 entries. We drop this column since it does not provide any useful information
- Pressure: In the pressure column there is 1288 values equal to 0. It is not possible to have zero air pressure on earth. We thus treat the 0 values as missing values.

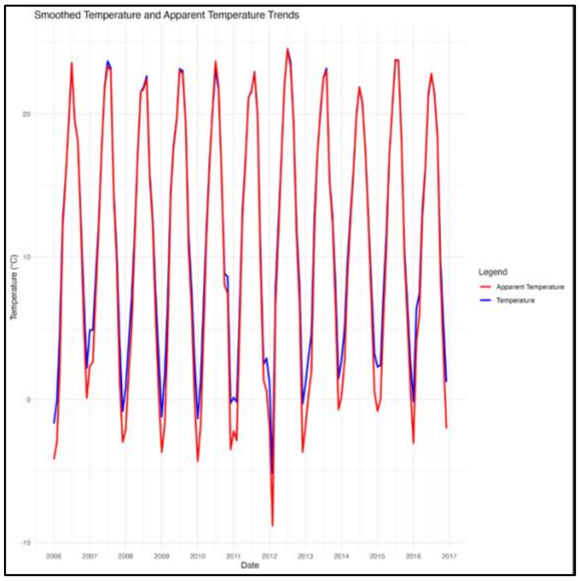


From the density plot above we see that it is reasonable to assume that the data follows a normal distribution for the purpose of imputing missing values with the EM-algorithm. Also, since the data is MAR, we can use the EM-algorithm to impute the missing values.

With the cleaned data, we can now look at the correlation between our numerical variables:



Furthermore, we plot Temperature and Apparent Temperature as a function of time.



The plot shows a clear annual cycle in temperature and apparent temperature, with peaks during summer and troughs in winter. While the trends in temperature and apparent temperature closely align, the apparent temperature on the lower ends are consistently lower than the real temperature. This could likely be due to factors like wind chill and humidity affecting perceived temperature.

2) Modeling

Modeling is done in Boosting_model.r.
We implement gradient boosting models to predict temperature and apparent temperature respectively.

2.1) Hyperparameters for boosting models

The hyperparameters of our boosting models are the following:

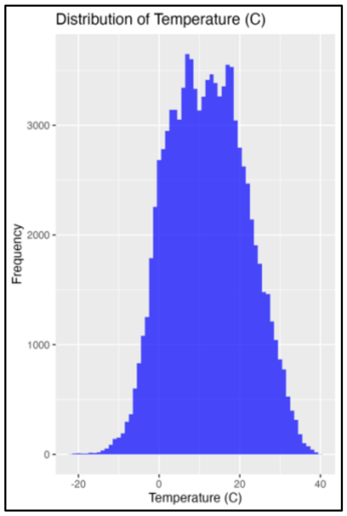
- Number of Trees (n.trees): The total number of trees to build in the model.
  - A higher number of trees can improve performance but increases computational cost and risk of overfitting.
- Interaction Depth (interaction.depth): The maximum depth of each decision tree.
  - Controls the complexity of the model. Deeper trees capture more intricate patterns but may overfit.
- Learning Rate (shrinkage): The step size at each iteration during boosting.
  - Smaller values make the model learn more slowly but generally result in better performance if combined with a higher number of trees.
- Minimum Observations in Node (n.minobsinnode): The minimum number of observations required in a terminal (leaf) node.
  - Helps control overfitting by preventing the tree from splitting too finely.
- Cross-Validation Folds (cv.folds): Number of folds used for cross-validation.
  - Determines how the dataset is divided for validation. For example, 5-Fold Cross-Validation, would mean that the dataset was divided into 5 folds, with 80% of the data used for training and 20% for validation in each iteration. More folds provide better evaluation but are computationally intensive.

We will outline the hyperparameter tuning process that was needed to find the optimal hyperparameters for the models.

2.2) Boosting model for Temperature prediction

2.2.1) Hyperparameter tuning

Before creating our boosting model, we plot the distribution of temperature.



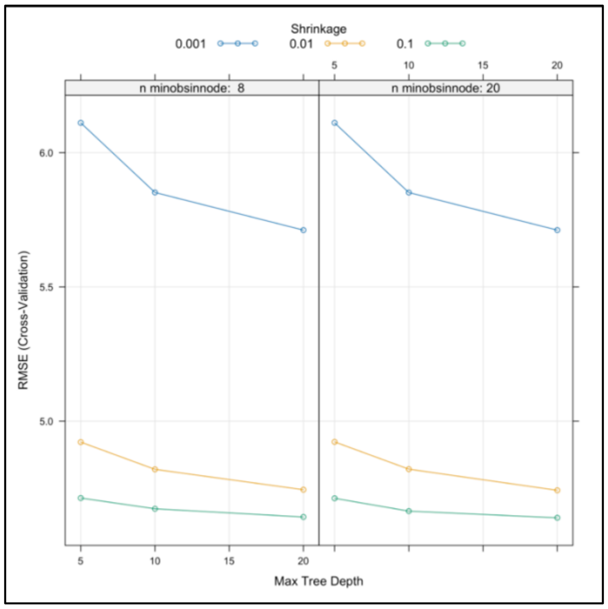
We see that it is reasonable to assume that temperature is normally distributed for our modeling purposes.

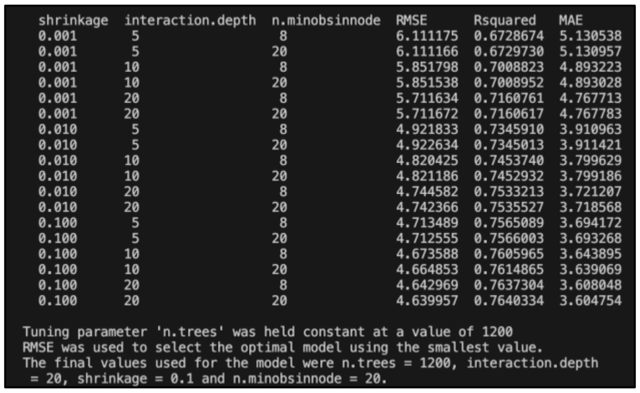
Before finalizing the hyperparameters for our model, we conduct a grid search to systematically explore the hyperparameter combinations of our model. This allows us to identify which parameters significantly influence performance and determine the optimal direction for fine-tuning them.

We define the following ranges to be explored by our grid search:

- interaction.depth: [5, 10, 20]
- shrinkage: [0.001, 0.01, 0.1]
- n.minobsinnode: [8, 20]

The model was trained multiple times, once for each combination of hyperparameters in the grid.





```
  shrinkage  interaction.depth  n.minobsinnode  RMSE      Rsquared   MAE
  0.001       5                  8              6.111175  0.6728674  5.130538
  0.001       5                 20              6.111166  0.6729730  5.130957
  0.001      10                  8              5.851798  0.7008823  4.893223
  0.001      10                 20              5.851538  0.7008952  4.893028
  0.001      20                  8              5.711634  0.7160761  4.767713
  0.001      20                 20              5.711672  0.7160617  4.767783
  0.010       5                  8              4.921833  0.7345910  3.910963
  0.010       5                 20              4.922634  0.7345013  3.911421
  0.010      10                  8              4.820425  0.7453740  3.799629
  0.010      10                 20              4.821186  0.7452932  3.799186
  0.010      20                  8              4.744582  0.7533213  3.721207
  0.010      20                 20              4.742366  0.7535527  3.718568
  0.100       5                  8              4.713489  0.7565089  3.694172
  0.100       5                 20              4.712555  0.7566003  3.693268
  0.100      10                  8              4.673588  0.7605965  3.643895
  0.100      10                 20              4.664853  0.7614865  3.639069
  0.100      20                  8              4.642969  0.7637304  3.608048
  0.100      20                 20              4.639957  0.7640334  3.604754

Tuning parameter 'n.trees' was held constant at a value of 1200
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 1200, interaction.depth
 = 20, shrinkage = 0.1 and n.minobsinnode = 20.
```

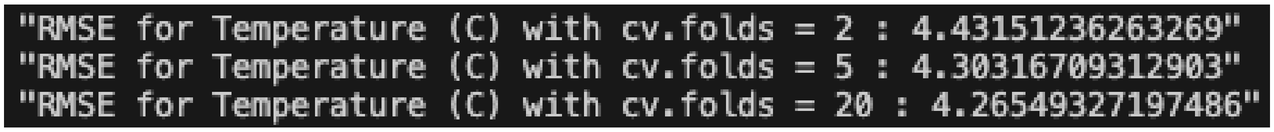
- Tree Depth and Complexity: Deeper trees (interaction depth of 20 compared to 5 and 10) lead to better performance, but at the cost of increased computational cost.

- Shrinkage: Higher learning rates (0.1) provided much better results than lower rates. A lower shrinkage (0.001) would require significantly more iterations to achieve comparable performance, making it less efficient for this problem.
- Minimum Observations in Nodes: Larger terminal nodes (n.minobsinnode = 20) provided slightly better results than smaller nodes (8).

From our grid search, we see that the best model has hyperparameters: Minimum Observations in Node =20, Max Tree depth =20, Shrinkage =0.1.

Note: Instead of just training our model on a 75%-25% train-test split, we train our data using cross validation because we observed a big improvement in performance, as we will show.

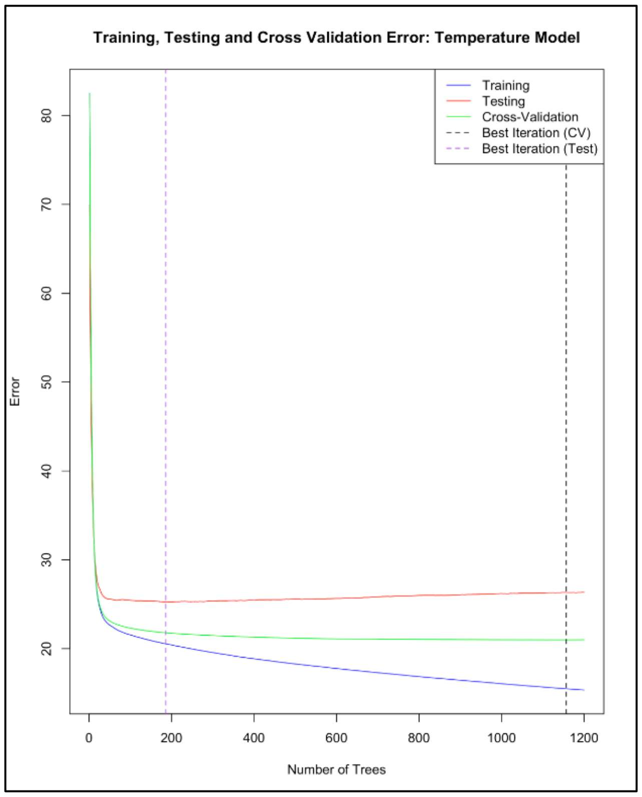
After identifying the optimal hyperparameters through grid search, three separate Gradient Boosting (GBM) models were trained using the best hyperparameters but with different numbers of cross-validation folds (cv.folds). This was done to assess how the number of folds affects model performance.



```
"RMSE for Temperature (C) with cv.folds = 2 : 4.43151236263269"
"RMSE for Temperature (C) with cv.folds = 5 : 4.30316709312903"
"RMSE for Temperature (C) with cv.folds = 20 : 4.26549327197486"
```

As the number of folds increases, RMSE decreases slightly, indicating better model performance. But the increased performance comes at a cost of increased computational costs.

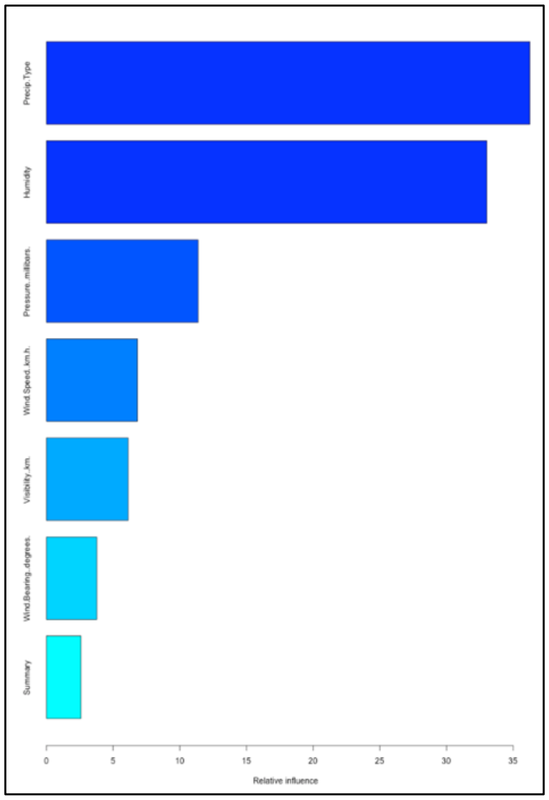
We illustrate the progression of training, testing, and cross-validation (CV) errors as the number of trees in the gradient boosting model increases.

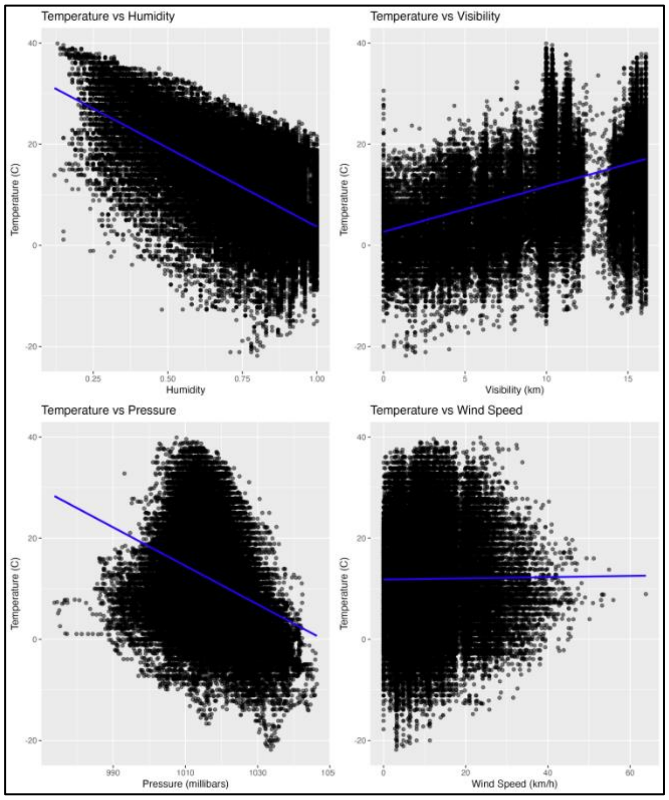


The training error consistently decreases, while the testing and CV errors initially decrease but begin to rise after their respective minima, indicating overfitting beyond a certain number of trees. The Best Iteration (Test) occurs at 186 trees, where the model achieves the lowest testing error for the 75%-25% train-test split, while the Best Iteration (CV) occurs at 1157 trees, reflecting optimal generalization across data subsets from cross validation. The CV-based model results in better out-of-sample predictions than the model with 75%-25% train-test split.

2.2.2) Feature importance

We plot the feature importance of our model to visualizes the relative contribution of each predictor variable to a model's predictions, highlighting which features most significantly impact the model's performance. We also plot the relationship between the most influential numerical variables and our predictor variable, temperature.





The feature importance plot and correlation plots provide complementary insights into how predictors influence the model. The correlation plots measure the linear relationship between predictors and temperature, while the feature importance plot reflects the contribution of each variable to the model's predictions, considering both linear and non-linear relationships.

The **precipitation type** is the most important predictor in our model. This shows its strong relationship with temperature. Snow indicate cold temperatures below 0 degrees while rain indicate temperatures above 0 degrees. In contrast, the **summary** variable is the least important predictor. Summary aggregates information from other variables already included in the model, making its contribution less distinct. This redundancy reduces its relative importance in the feature importance plot. The model essentially prioritizes variables that provide unique and non-redundant information, which explains the observed difference in importance between these two factor variables.

The scatterplots reveal that relative **humidity** has a strong negative relationship with temperature. Relative humidity is the percentage of water vapor in the air compared to the maximum amount the air can hold at a given temperature. Colder air has a lower capacity to hold water vapor compared to warmer air. Thus, if the absolute amount of water vapor in the air (absolute humidity) remains constant, the relative humidity increases as the temperature drops because the air's capacity to hold water vapor decreases. Hence the negative relationship that the scatter plot reveals. The relative influence of humidity on temperature suggests that the model captures this linear relationship, non-linear effects, and additional interactions between humidity and other variables.

In the earth's atmosphere, the expected relationship between temperature and **pressure** works as follows: As temperature increases, warm air becomes less dense (molecules spread out), causing it to rise. This can lead to lower surface pressure because the rising air reduces the mass of air pressing down on the surface. Conversely, colder air is denser and tends to sink, leading to higher surface pressure. In the scatter plot we can clearly see the inverse relationship which we would expect. Pressure also appears as an important feature in the model, suggesting that the model is probably also capturing non-linear and interaction effects.

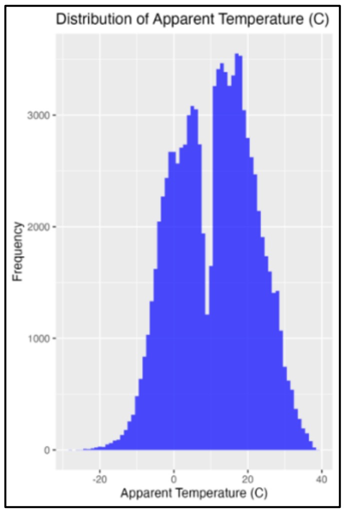
Wind speed shows a near-zero linear correlation in the scatterplot but is moderately important in the feature importance plot. This indicates that wind speed contributes through complex, non-linear effects and interactions.

Visibility shows a positive correlation with temperature, as clearer skies are often associated with warmer weather.

2.3) Boosting model for Apparent Temperature prediction

2.3.1) Hyperparameter tuning

we plot the distribution of apparent temperature.

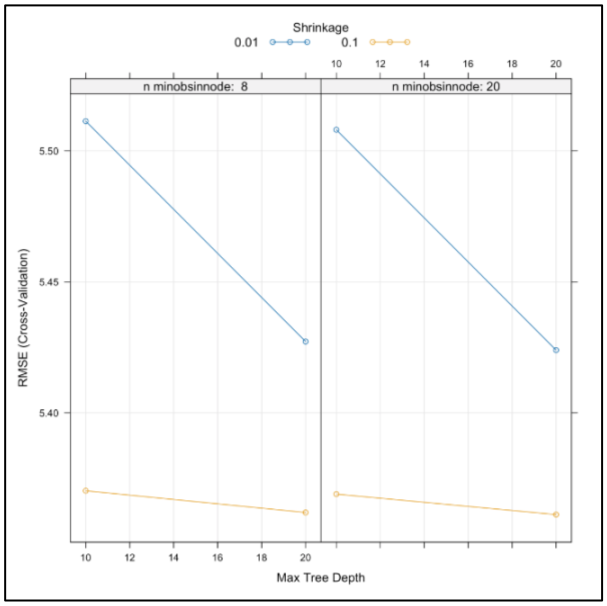


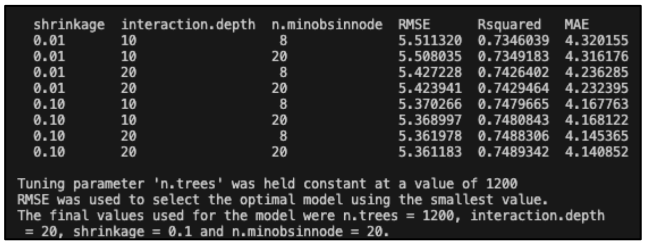
We can assume that the apparent temperature is normally distributed for the model. Gradient boosting is robust to some deviations from normality, even for bimodal distributions as seen here.

For the apparent temperature model, we will again conduct a grid search as before. But this time we will do a smaller grid search just to verify that we get similar results for the optimal hyperparameters as for the temperature model.

We define the following ranges to be explored by our grid search:

- interaction.depth: [10, 20]
- shrinkage: [0.01, 0.1]
- n.minobsinnode: [8, 20]



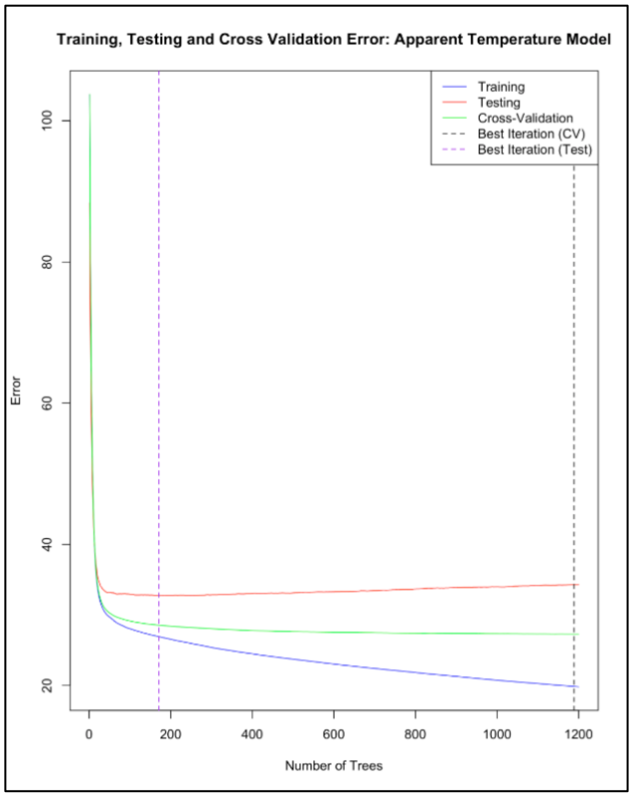


```
   shrinkage  interaction.depth  n.minobsinnode  RMSE      Rsquared   MAE
   0.01       10                  8              5.511320  0.7346039  4.320155
   0.01       10                 20              5.508035  0.7349183  4.316176
   0.01       20                  8              5.427228  0.7426402  4.236285
   0.01       20                 20              5.423941  0.7429464  4.232395
   0.10       10                  8              5.370266  0.7479665  4.167763
   0.10       10                 20              5.368997  0.7480843  4.168122
   0.10       20                  8              5.361978  0.7488306  4.145365
   0.10       20                 20              5.361183  0.7489342  4.140852

 Tuning parameter 'n.trees' was held constant at a value of 1200
 RMSE was used to select the optimal model using the smallest value.
 The final values used for the model were n.trees = 1200, interaction.depth
  = 20, shrinkage = 0.1 and n.minobsinnode = 20.
```

From the grid search we get the exact same pattern as before. We see that the best model again has hyperparameters: Minimum Observations in Node =20, Max Tree depth =20, Shrinkage =0.1.

After seeing how the number of cross-validation folds (cv.folds) impacted model performance before, we will now train only one model with a relatively large number of cross-validation folds (20).

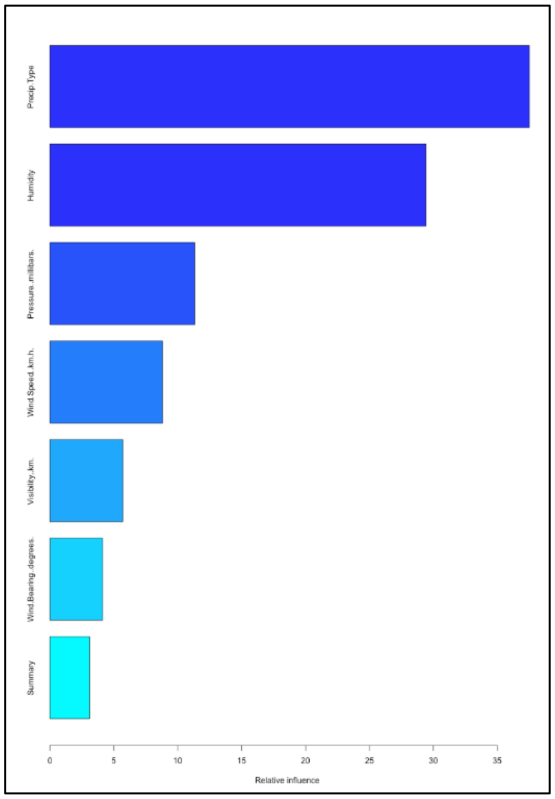
We illustrate the progression of training, testing, and cross-validation (CV) errors as the number of trees in the gradient boosting model increases.

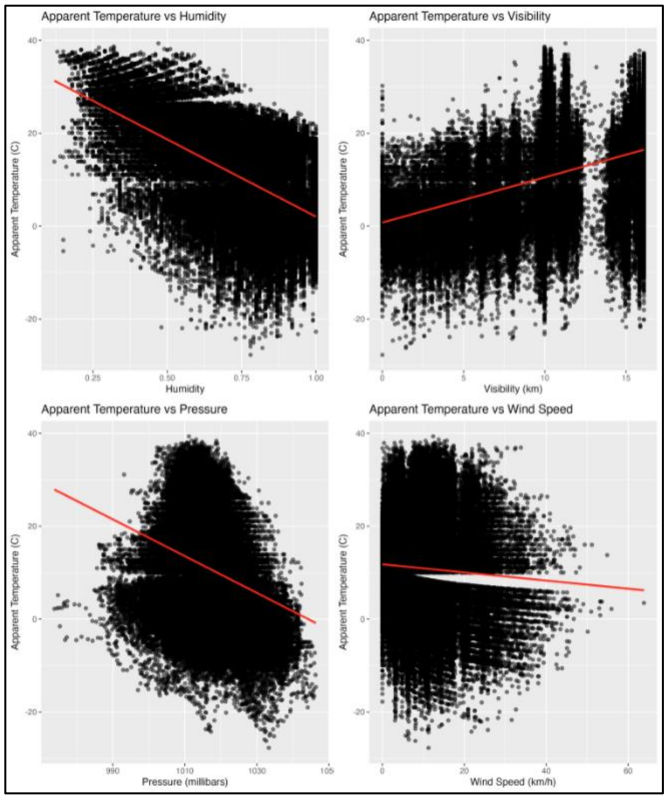


The training error consistently decreases, while the testing and CV errors initially decrease but begin to rise after their respective minima. The Best Iteration (Test) occurs at 171 trees, where the model achieves the lowest testing error for the 75%-25% train-test split, while the Best Iteration (CV) occurs at 1189 trees, reflecting optimal generalization across data subsets from cross validation. The CV-based model results in better out-of-sample predictions than the model with 75%-25% train-test split.

2.3.2) Feature importance

As before, for the temperature boosting model, we plot the relative importance of the features and the relationship between our dependent variables and apparent temperature.





The feature importance plot and correlation plots for the apparent temperature model can be interpreted in the exact same way as for the temperature model.

The only difference is that we see that the relationship between **wind speed** and apparent temperature differs significantly from that with temperature due to how wind influences perceived conditions versus actual atmospheric measurements. The scatterplot for temperature and wind speed showed a near-zero correlation, indicating that wind speed has little direct effect on actual temperature measurements. However, the scatterplot for apparent temperature and wind speed shows a stronger negative correlation. Higher wind speeds reduce the apparent temperature due to the wind chill effect. The wind chill effect occurs because wind accelerates the loss of heat from the skin, making it feel colder than the actual air temperature, especially at low temperatures. This negative correlation is incorporated into the model and can be seen from the higher relative influence of wind chill on the apparent temperature model than on the temperature model.

Conclusion

We implemented gradient boosting models to predict temperature and apparent temperature for the city of Szeged.

The data cleaning process addressed anomalies, missing values, and redundant variables.

For both temperature and apparent temperature models, hyperparameter tuning through grid search identified optimal values for tree depth, shrinkage rate, and minimum observations per node. We showed that cross-validation increases model performance compared to a standard 75%-25% train-test split of the data. Furthermore, a higher number of cross-validation folds further increases model performance (at the cost of higher computing costs).

The analysis demonstrated that precipitation type, humidity, and pressure were the most influential predictors, and that summary and wind bearing degrees were the least influential.

A notable difference was observed in how wind speed influenced temperature compared to apparent temperature. While wind speed showed negligible correlation with temperature, its negative relationship with apparent temperature showed the wind chill effect on perceived weather conditions.

The results demonstrate the successful usage of gradient boosting for weather prediction.