Instituto Superior Técnico

Advanced Automation

MEMec

Hardware: Weather Forecasting with Arduino Temperature and Luminance Sensors

Group 13

Elliot Alexander Ferning, 105003
Hallvard Bjørgen, 105243
Manuel Sach, 105357
Miguel Cruz Irimia, 104537

Professors: João Sousa, João Santos

Fall 2022

Project Proposal Group 13

The aim is to collect data for temperature and light using sensors connected to an Arduino microcontroller. The Arduino will read from it's sensors, and should be able to output the forecast for the next two hours. This will be done by fitting (most likely) a linear model to the data, which will acquired from the Arduino for (most likely) four consecutive days at the same hours each day. Should there be a need for more data to properly train the model, this data will be collected from an external resource using an API.

The Arduino will also be used to measure the actual temperature and light after the data-collection period in order to validate the model.

Some of the **main objectives** we are interested in seeing is
(a) if the temperature for a given day can be (to some degree) forecast given the way it's been some previous days and the hours leading up to the moment of making the forecast, and
(b) if the model created will predict that the weather will still be cloudy given that there's been clouds thus far the given day, and
(c) if there will be a discrepancy in the data measured from the different sensors (we are hoping to use 2 or 3 of both types).

The **necessary hardware will be** 1 Arduino Uno kit (with breadbord, wires and resistors), preferably with 3 temperature sensors and 3 luminance sensors.

The group is also keen to explore ChatGPT3's possibilities

ChatGPT3 is a efficient tool and some professors at IST are arguing that the programming skill might somewhat reduce to knowing how to use the AI tools to write good code rather than actually writing the code oneself, due to its superior efficiency.

Including this in our project, we could make the chatGPT3 write all our code for models and do most of the data preprocessing, leaving us with the more interesting project of evaluating and commenting on the usability and easiness of using chatGPT3 in our project, and support for it's theoretical answers on questions related to for example what are the best ways to forecast temperature values etc.

We'd still make the hardware job of setting up the arduino with sensors and reading the data and training the models with it, but instead of using a linear regression model, we'd implement multiple models by having chatGPT3 write us the code and providing some of the analysis. We could then, rather than fine tuning hyperparameters and such, go off the course syllabus and focus on how the chatGPT3 is performing in doing a data scientists job.

Whether the "chatGPT3 exploration"-part is added to the project or not depends on what the professors gives as feedback to this project proposal, and as it's unclear whether this would increase or decrease the workload, it will also depend on how much time the students have for the project when time comes to do the project.

1 Data collection using an arduino

Hardware:
1) Arduino UNO
2) Wires, cables and sensors
3) Arduino MKR WiFi 1010 (with ESP32 module)
4) Micro USB cable

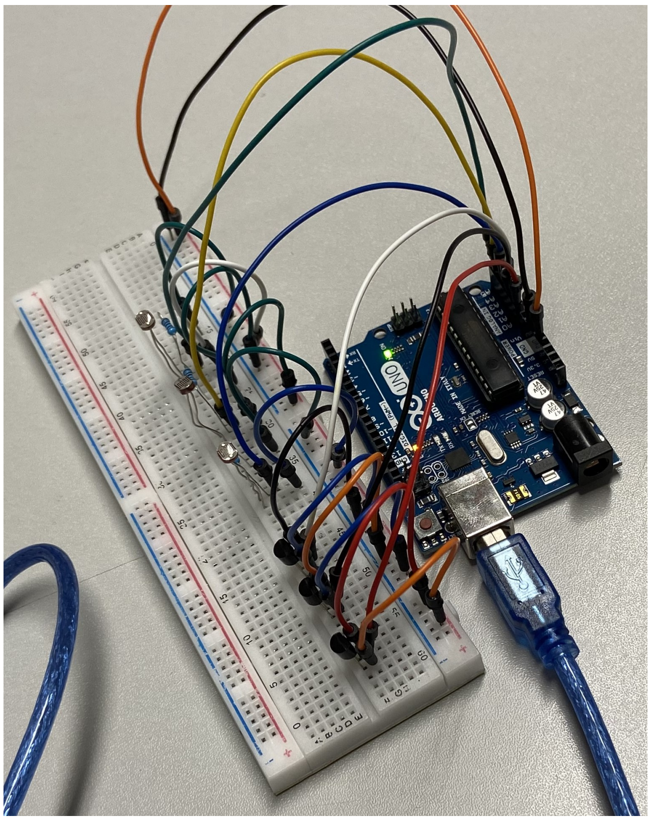


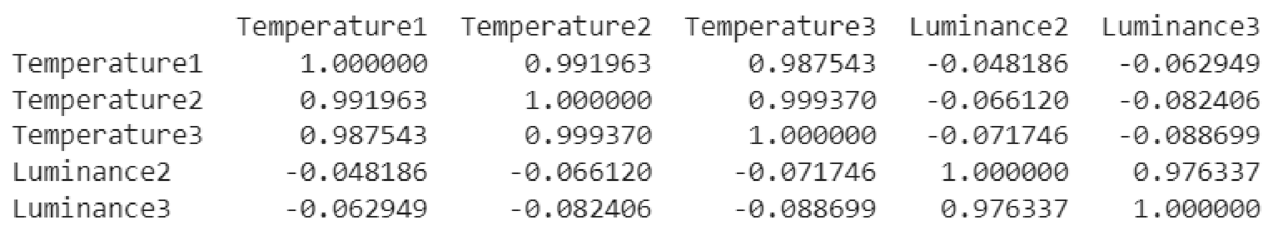
Figure 1: The setup of the Arduino with 3 temperature sensors and 3 luminance sensors.

In order to collect temperature and luminance data through the Arduino UNO, the microcontroller board has to be connected to the computer while the data collection is running. In order to get sufficient amount of data to train models and compare their results, the board would have needed to be connected to a computer all day outside. Having other tasks to do in a day and not wanting to leave the computer outside by itself all day, a way to transmit the data over the internet then quickly became a critical need for the data collection process to be feasible for this project.

We got an MKR WIFI 1010 board to connect to the internet which hopefully, according to the different tutorials we read [1],[2], would allow us to
(1) host a server on the MKR board,
(2) send data to this server, and
(3) have our computer remotely access this data as a client.
This sounds an awful lot like setting up an API, which is exactly what was needed for this project.

All code was set up, a micro-USB cable was obtained, and all was ready to upload the code to the board, but the micro-USB port on the board was dysfunctional (we believe, despite being brand new) so the work in this regard was stopped and the plan then turned into using an external API to train the model and then hopefully using Arduino sensor data from a couple of hours on the finished models to forecast the two next hours.

As we saw from the sensors readings, the temperatures were approximately reading the same values, while the luminance sensor 1 had some obvious error. The failed sensor was tried to be replaced with three other sensors and also the wiring on the circuit board was tried to be moved. The fault could be in the analog input from the arduino UNO where it registers its analog sensor reading. The idea was to use luminance sensor readings as a way to see if it has some correlation with the temperature. However, it does not seem to be correlated at all, which can be due to various reasons as seen in figure 2.



```
              Temperature1  Temperature2  Temperature3  Luminance2  Luminance3
Temperature1      1.000000      0.991963      0.987543   -0.048186   -0.062949
Temperature2      0.991963      1.000000      0.999370   -0.066120   -0.082406
Temperature3      0.987543      0.999370      1.000000   -0.071746   -0.088699
Luminance2       -0.048186     -0.066120     -0.071746    1.000000    0.976337
Luminance3       -0.062949     -0.082406     -0.088699    0.976337    1.000000
```

Figure 2: Correlation matrix between temperature and luminance data.

There are also several observations where the sensor would gain a sudden peak or a drop, which could perhaps be seen as outliers, see figure 13,4,5.

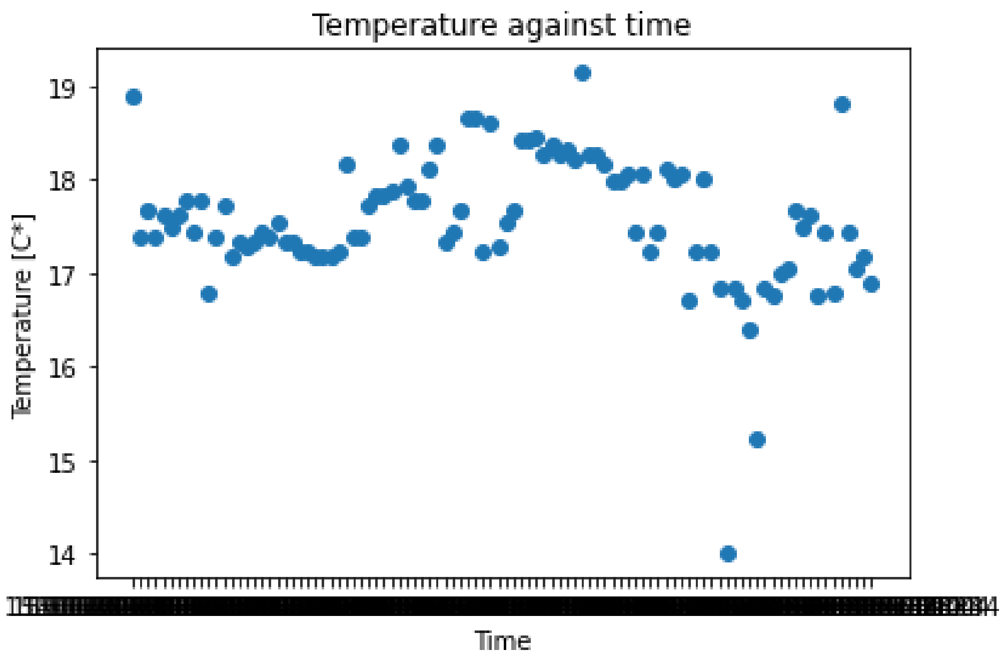


Figure 3: Scatter plot of temperature against time for sensor readings on Sunday

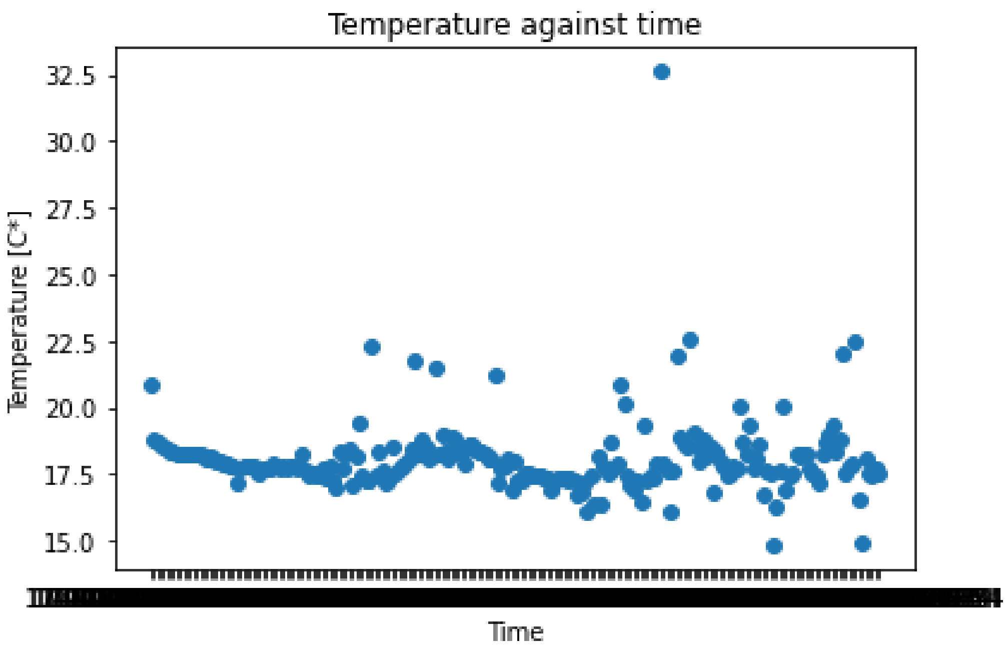


Figure 4: Scatter plot of temperature against time for sensor readings on Tuesday

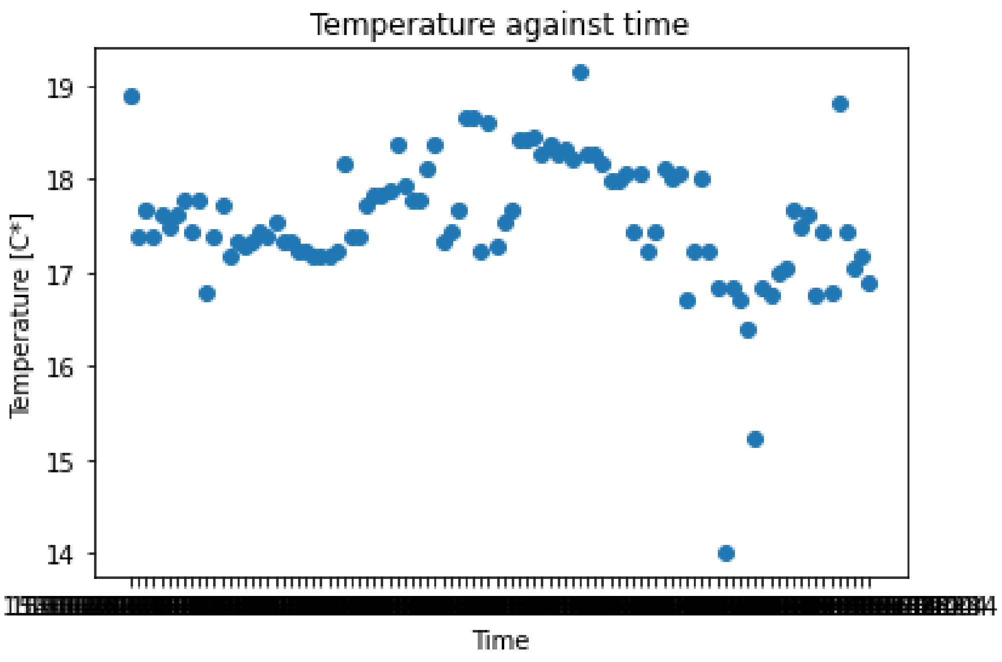


Figure 5: Scatter plot of temperature against time for sensor readings on Wednesday

2 Data pre-processing

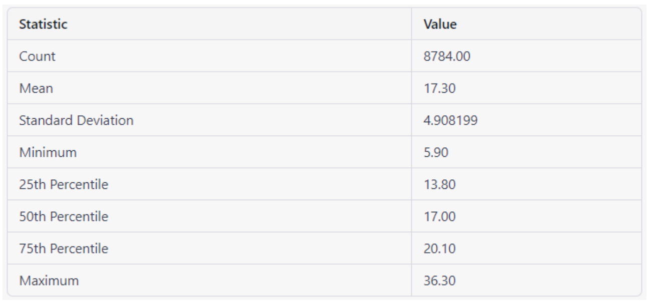
Data preprocessing can help filter out irregularities and noisy data in the dataset. We start by getting an overview of the data and identifying any missing values.

As previously mentioned the sensor data collected through the arduino would not be sufficient for the project. The solution was to integrate an API that collects historical weather data from an open source database called open-meteo [3] which contain 60 years of data. The main data interested was the hourly temperature in Lisbon which is retrieved and stored in variables later to be used.

To evaluate the temperature data obtained from a single API, the first thing to do is to understand the data values. The temperature data is represented in degrees Celsius, which means that the values range from 0°C to 100°C. This can cause issues in terms of classification, as the data is limited to a very small range of values. Therefore, it is necessary to normalize the data to ensure that the machine learning models are able to process the data correctly. This will help to improve the quality of the data and ensure that the machine learning models are able to classify the data with greater accuracy. Additionally, these technique will also help to reduce the training time and improve the processing speed. As we only use one API, we don't have to worry about the unification of different datasets or error detection and removal of redundant or incomplete information.

Now, we decide to make a exploratory data analysis (EDA), that involves the use of visual tools to understand the distribution of data and the relationships between variables.

First we see the characteristics of the temperature data by using the info() and describe() functions. We obtain the following statistics:



Statistic	Value
Count	8784.00
Mean	17.30
Standard Deviation	4.908199
Minimum	5.90
25th Percentile	13.80
50th Percentile	17.00
75th Percentile	20.10
Maximum	36.30

Figure 6: Dataset characteristics.

Some ideas we can extract from this that most of the temperature observations are within a range of 13.8 C to 20.1 C, which is the interquartile range (IQR). This suggests that the data is relatively consistent with only a few extreme value. Also, the range of temperature is quite large and the standard deviation is relatively high. In summary, the dataset of temperature by hour in one year has a mean temperature of 17.30 C, median of 17.00 C and a standard deviation of 4.9 C. The data is relatively consistent with only a few extreme values. The range of temperature is quite large and the standard deviation is relatively high, which confirm that the data is quite variable and spread out over a wide range of values. With the histogram of the temperature data and the graph of the temperature data over time we can get the idea of the distribution of temperatures throughout the year, such as whether the temperatures are generally high or low.

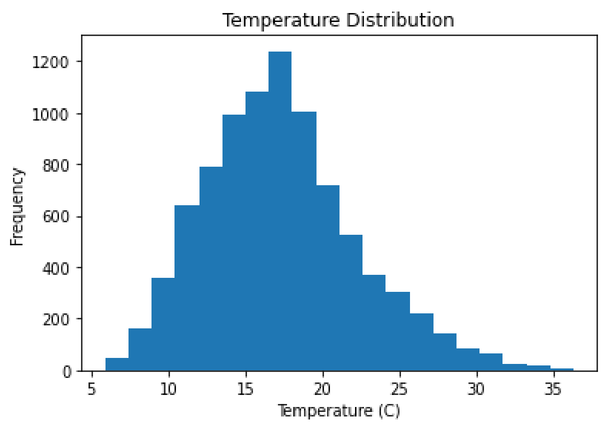


Figure 7: Distribution of temperatures throughout the year.

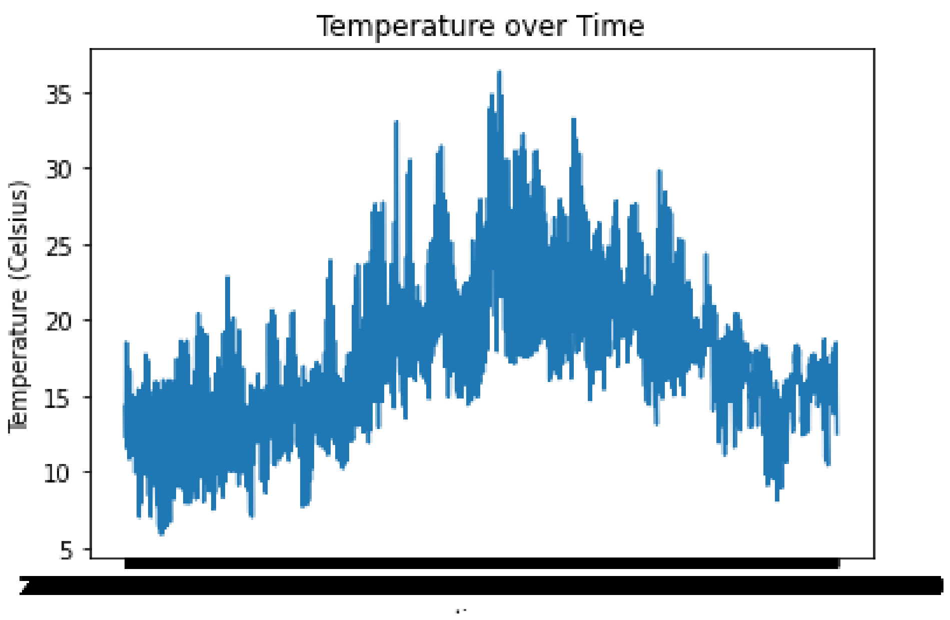


Figure 8: Variation of temperature data over time.

Other way to see the temperature data over time is the line chart of the temperature data that show the temperatures vary throughout the year, such as if there are any patterns or trends in the data, and if there are any notable periods of high or low temperatures.

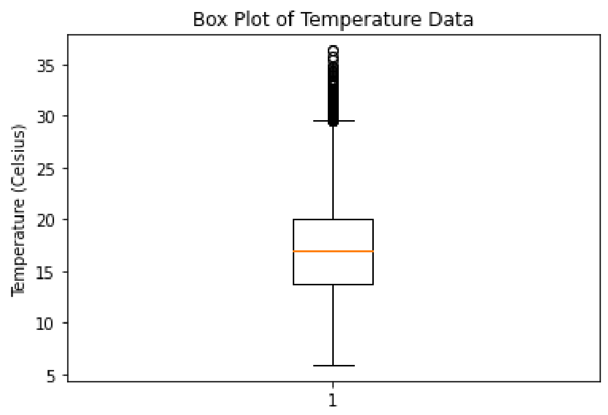


Figure 9: Box plot of the temperature.

Finally, we group the temperature data by month and calculate the mean temperature for each month to plot the mean temperature by month and get a visual idea of how the temperatures vary by season and how they compare with each other.

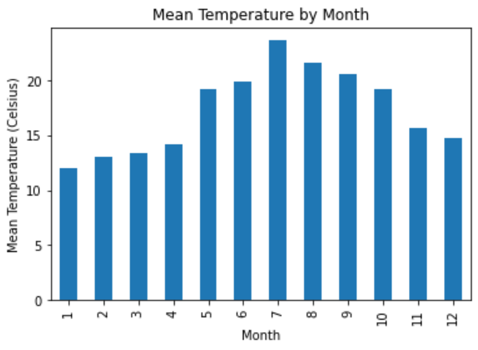
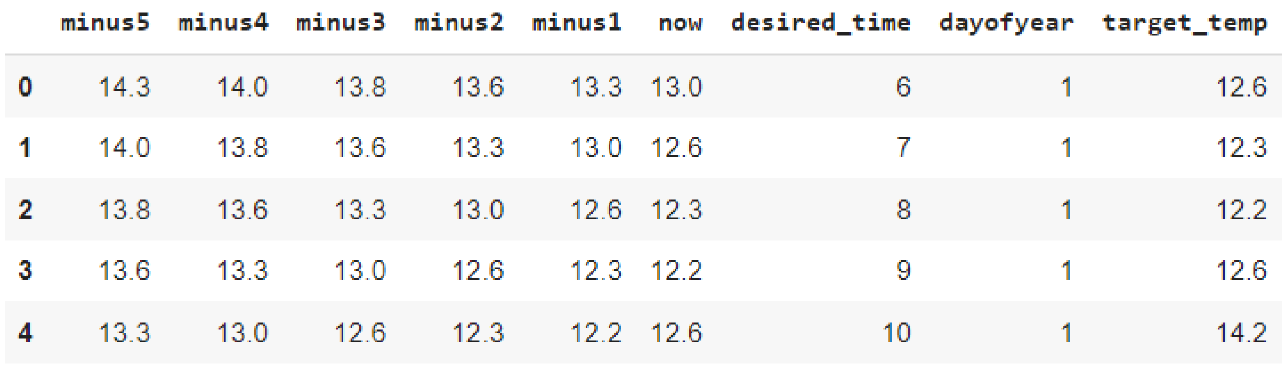


Figure 10: Average temperature by month.

3 Creating the training dataframe

For forecasting the temperature it is mandatory to define the predictors and the target values. In this case the target values are the temperatures for the next two hours. At first the focus is on the forecast for the temperature one hour in the future. The target value is dependant on the past temperatures, the time of day and the time of the year. Therefore a dataframe has to be created that contains all that information, which is shown in 11.



	minus5	minus4	minus3	minus2	minus1	now	desired_time	dayofyear	target_temp
0	14.3	14.0	13.8	13.6	13.3	13.0	6	1	12.6
1	14.0	13.8	13.6	13.3	13.0	12.6	7	1	12.3
2	13.8	13.6	13.3	13.0	12.6	12.3	8	1	12.2
3	13.6	13.3	13.0	12.6	12.3	12.2	9	1	12.6
4	13.3	13.0	12.6	12.3	12.2	12.6	10	1	14.2

Figure 11: Head of Dataframe for Training

The dataframe contains the 'desired_time' which is the time at which the temperature should be forecast. With the 'datetime' package it is possible to extract from the timestamp on which day of the year the data was collected. With the specification 'timetuple().tm_yday' an Integer from 1 to 365, or 366 for leap years, is produced. By doing that seasons are taken into account. For the past temperatures 6 data-points are used which provides enough data points for the forecast but not too many so that computing power doesnt get too high. The number of past data-points can be adjusted by experimenting but would exceed this assignment for now.

4 Models

When building a predictive model, it is important to try different types of models and compare their performance in order to choose the best one for the specific problem. This process is commonly referred to as model selection or model comparison.

In this case, we will be comparing several models for forecasting temperature data by hour in one year. Models evaluated are linear regression, support vector regression (SVR), and neural network (NN) models.

To compare the performance of these different models we are using R^2 score. We decide to use it because is a measure of the goodness of fit of the model that tells us how well the model's predictions fit the true values and is frequently used in the field of machine learning and statistics because it is a well-established and widely accepted evaluation metric for comparing models.

Additionally, it's important to keep in mind that the architecture and the parameters of the model can have a significant impact on the model's performance, so it's necessary to try different ones in each modl.

4.1 Linear regression

Since Temperature Forecasting in short term is mostly a linear model, the simplest form is the Linear Regression. But the temperatures are rising and falling cyclic over a day and the mean temperature is cyclic over the year. So at night most of the time its colder than at day and in the summer colder than in the winter. Therefore the package 'PolynomialFeatures' is used to mirror that behaviour while still being able to do a Linear Regression. With the data obtained and prepared as described in the section 'Creation of Training Dataframe' the model of the linear regression was trained. With the 'random' package a random seed was created to split the data set into a training and a test set. The test set is 30% of the whole data. In the first iteration only data of the first 2 weeks of January 2023 was used to train and test the model. In this case the time of the year wasnt taken into account because all the data was in the same season. For this dataset models with and without PolynomialFeatures were compared. In this case with a low amount of data there is almost no difference in R^2 Score but without it it is a bit lower(0.997 and 0.995). That is why we used the PolynomialFeatures for the following models. Also it makes sense as stated in the beginning of this section because a polynom can mirror a temperature over the time of a day or year better. After that we used the whole year of 2022 to train the model and took into account which day of the year it is. This was done with the 'datetime' package as described earlier. With the data of a whole year the LinearRegression achieved a R^2 score of 0.994. In this case the PolynomialFetures were also used. Overall it is obvious that the model of the Linear Regression can achieve really good results but it will be compared to other models in the next sections.

4.2 Support Vector Regression

Support Vector Regression (SVR) is a type of support vector machine (SVM) that can be used for regression tasks. Like other SVM models, SVR is a powerful and flexible algorithm that can be used for various regression tasks, including forecasting.

SVR does require careful tuning of the model's parameters, particularly the kernel function and regularization parameter, to achieve good results. Also, as SVR is a non-parametric model, it can be more computationally expensive than other linear regression models. The most appropriate kernel will depend on the specific characteristics of the time series data. We compare the following ones: radial basis function kernel (a non-linear kernel function that can model a wide range of non-linear relationships) polynomial kernel (which can model non-linear relationships by fitting polynomial functions to the data) and linear kernel (which performs linear regression, useful when the data has a linear relationship).

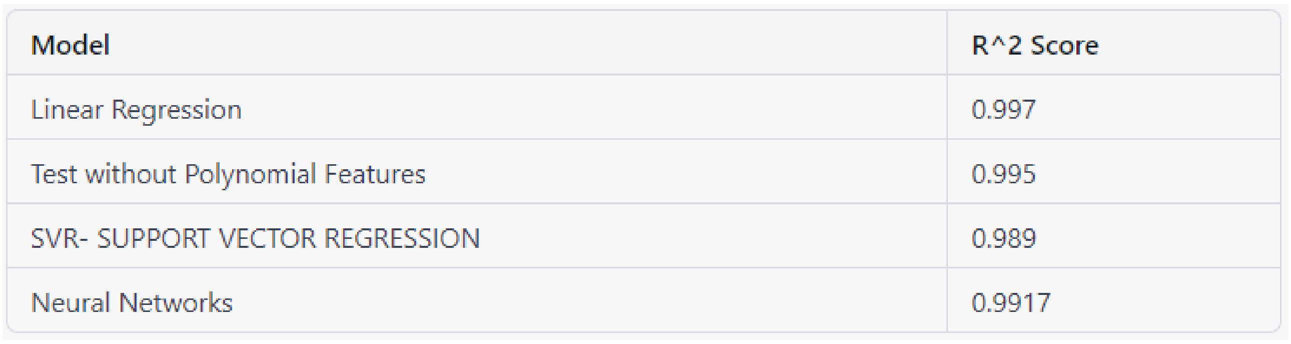
We get the best result with the polynomial kernel.

4.3 Neuronal Networks

A neural network (NN) can also be used for forecasting the temperature data by the hour in one year by learning patterns and relationships in the data. In this process we define different architectures and parameters to find the best model for the specific dataset. This involves finding the best number of layers, the number of neurons in each layer, and the activation functions to be used.

After trying different configurations of parameters, testing ReLU (Rectified Linear Unit), Sigmoid and tanh (Hyperbolic Tangent) activation functions, we get best results wirh ReLu, a simple and commonly used activation function in neural networks. It returns the input if it is positive, and returns 0 if it is negative. It is computationally efficient and helps to prevent the vanishing gradient problem, which is a common issue in deep neural networks. In this model, after trying different configurations of hidden layers and neurons we get the best result with 3 hidden layers and 128 neurons. So that's the neuronal network model we will use to compare with the others.

4.4 Results



Model	R^2 Score
Linear Regression	0.997
Test without Polynomial Features	0.995
SVR- SUPPORT VECTOR REGRESSION	0.989
Neural Networks	0.9917

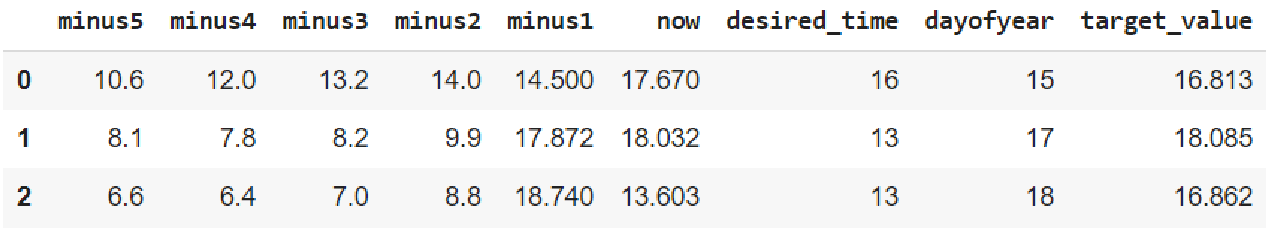
Figure 12: Comparation between models

The table shows the R^2 scores for different models. It can be seen that the linear regression model has the highest R^2 score of 0.997, indicating that it has the best fit to the data among the models tested. The test without polynomial features has a slightly lower R^2 score of 0.995, while the SVR has a lower R^2 score of 0.989. Lastly, the Neural Network model has a R^2 score of 0.9917.

Since all models scored highly in the evaluation criteria, it is desirable to choose a simple model which can achieve the same level of prediction accuracy. In this case, a linear regression model would be best suited for predicting the weather. Since it only has temperatures for different hours as predictors it is fairly a simple model and behaves linearly. If other predictors were to be included it could add to the complexity of the weather prediction and thus other machine learning models would be more suitable.

5 Using the Arduino data to predict the next hour

In order to try to use the Arduino data to get a prediction, we structure the available data so that it can be used as hourly predictors which is a maximum of 2 hours. The rest of the data is filled in with the API data. This data act as a test set that is going to be run with the previously trained models. These models are as previously explained trained on the API data. For this purpose, we only use the linear regression model to see how well it predicts the next hour's temperature since it is the easiest model and gives the best result.



	minus5	minus4	minus3	minus2	minus1	now	desired_time	dayofyear	target_value
0	10.6	12.0	13.2	14.0	14.500	17.670	16	15	16.813
1	8.1	7.8	8.2	9.9	17.872	18.032	13	17	18.085
2	6.6	6.4	7.0	8.8	18.740	13.603	13	18	16.862

Figure 13: Test data set manually constructed to be tested on the trained linear regression model.

The resulting prediction for the next hour was 20.95603672, 18.1308047, 15.46890832 which is a poor prediction and thus yield a bad r^2 score. One reason for this is that temperature sensor readings usually show a higher temperature than compared with the API data. This could be because of the sun heating up the temperature sensor to above air temperature. Ideally, the temperature sensors should not have had direct sun exposure which contributes to a higher temperature reading.

6 Conclusion

Due to limited time and the need for manual collection, only a total of 5 hours of data was collected during a span of 3 days. The lack of hourly data and consistency proves it to be difficult to make any predictions, even with a well-trained model using API historical data. The use of three sensors had some value in that if one or two sensors fail, the other can show reasonable results. It was noticed the temperature sensor readings usually show a higher temperature than it was on that day and time and compared with the API data. This could be because of the sun heating up the temperature sensor to above air temperature. Ideally, it should have been covered from direct sun exposure. This is also why the Arduino test results aren't good. Due to the jumps in temperature from the API data to the arduino data the model does not work well. In the final work of the project, we also noticed that the data frame used from the API to fill the hours was loaded from year 2022 and that it did not exist any data for the last two weeks in 2023. This can also be one cause for why the result is not very reliable.

The scope of the project could have been narrowed down to focus on a fewer set of models and put a higher focus on them. The project proposal had ideas that we wanted to investigate, but due to lack of time, we decided to abandon it. For example, investigating about how chat GPT3s ability to write code and how it can affect the future of data science.

References

[1] *Connecting MKR WiFi 1010 to a Wi-Fi network*, `https://docs.arduino.cc/tutorials/mkr-wifi-1010/connecting-to-wifi-network`, Accessed: 2023-1-20.

[2] *Host a web server on the MKR WiFi 1010*, `https://docs.arduino.cc/tutorials/mkr-wifi-1010/hosting-a-webserver`, Accessed: 2023-1-20.

[3] *Historical weather API*, en, `https://open-meteo.com/en/docs/historical-weather-api`, Accessed: 2023-1-20.