2023

Modelling And Optimization Under Uncertainty

12135637
ROSHAN DHANASEKARAN

SUBJECT CODE: &135CEM

Temperature And Climate Change Prediction Using Gaussian Processes Regression

Dhanasekaran, Roshan

12135637
Faculty of Engineering, Environment, and Computing
MSc Data Science and Computational Intelligence, Coventry University
dhanasekar@uni.coventry.ac.uk

Abstract – Since the earth's temperature and climate have a stochastic nature (Chen, Qian, Nabney & Meng, 2014), precise and accurate temperature prediction can be sometimes intriguing this is mainly because there are many factors which lead to temperature and climate change. In this paper, we intend to use a probabilistic model: Gaussian Processes (GP), and with the help of GP, we can find the uncertainties in our predictions and get a probability estimate of how the temperature and weather can vary in the future. In this paper minimum temperature, maximum temperature, Rainfall, wind speed and humidity will be considered as variables for prediction since they are some of the major contributors affecting weather change. We only consider a single station's data which is in Cobar, Australia to perform our tasks. The dataset used in this paper for training and testing was taken from Kaggle called rain in Australia. We create three models with different covariant matrix then compare and validate the models' empirical outputs based on the evaluation metrics. The paper also discusses how the kernel/covariates metrics can affect the models' outputs and predictions. The results show that the Exponential sign square kernel was able to predict and find uncertainties far in the future but although the Radial basis function (RBF) was good with predictions it failed to predict data far in the future. These models and kernels were particularly chosen to tackle the problem of uncertainty the prediction can also be performed with a simple RNN network, but the RNN does not provide us with the uncertainty measure therefore using gaussian processes was the right method to perform the tasks.

Keywords – Gaussian Processes, uncertainty, mean square error, Radial basis Function Exponential sign square, Root mean square

I. INTRODUCTION

Developing a robust weather prediction model is an absolute necessity now more than ever with the increase in global temperature of 0.08 Celsius per decade and the alarming increase in sea levels by 21-24 centimetres since 1880 has affected this planet and all the habitats living in it, with that said we as humans have the responsibility to protect the natural environment for our future generations and make the earth a better place to live. Considering the recent natural disasters that have occurred in recent times like the Australian forest fire and the flash floods in Indonesia can be related to climate change and this issue soon needs to be addressed.

A study taken with the advancements of Operational centers achieved significant improvements in the accuracy of global numerical weather forecasts throughout and after the Global Weather Experiment in 1979. They are the outcome of advancements in the worldwide observing system, practical analysis and modelling study, and escalating computing capacity in operational prediction (Kalnay, Kanamitsu & Baker, 1990). Another study in 2014 took place in China where wind power was forecasted using Gaussian processes that have significantly helped wind farms with calculating wind power units and energy reserve scheduling. A study taken place in Australia on predicting Maddon jillion oscillation(MJO) rather than considering individual variables like temperature, humidity and windspeed the study has predicted the location of the MJO at various times of the year which intern can tell us the temperature change and take necessary precautionary measures to prevent any types of disaster (Seo et al., 2009).

The remainder of this paper is organised as follows. Section 2 describes the problem statement and the problem which we are trying to address. Section 2 also contains the description of the dataset which is utilised in performing regression. Section 3 covers the technical aspect of the paper by first describing gaussian processes and discussing the kernels used in performing regression. Section 4 presents an overview of how this experiment was carried out and the packages used and the

necessary pre-processing steps involved to get the final results. Section 5 covers the outputs which we got on performing prediction and the outputs the advantages and disadvantages of the model were discussed. In section 6 we discuss in the depth the outputs and if we have addressed the problem statement and finally discuss if there are any legal and ethical issues on performing this experiment followed by the conclusion.

.

II. PROBLEM STATEMENT / DATASET

The vision behind performing this task is to have better living conditions, improve weather forecasts, logical decisions making and implement sustainable projects for the future. the problem we are trying to solve is predicting the uncertainties in weather conditions in a single location in Cobar, Australia the outcomes of this paper can be utilized with all the major cities in the world which can intern benefit this planet. Addressing this problem can have an impact on businesses, farming, and the production industry. For example, on determining the temperature rise in a certain location we can find the cause of the rise in temperature if the cause was excess carbon emission the government can take necessary steps to reduce carbon footprints in a certain area which can lead to a reduction in temperature. According to NOAA's 2021 Annual Climate Report, the combined land and ocean temperature has increased at an average rate of 0.14 degrees Fahrenheit (0.08 degrees Celsius) per decade since 1880 (Climate Change: Global Temperature, 2022). Addressing global warming has been very important now more than ever, on performing predictions and finding uncertainties for every location in the world we can overall address the global warming behemoth.

The dataset which was used to perform all the tasks was taken from Kaggle but originated from the Australian bureau of meteorology. The dataset contains 22 columns and 56420 rows, although not all the rows and columns were utilised to perform this task one of the reasons being computational power and we only use minimum temperature, maximum temperature, date, rainfall, windspeed and humidity and the

experiment was conducted only for 177 days and future predictions were made by adding extra 50 days.

Table 1 Dataset description

VARIABLE NAME	VARIABLE FORMAT	DESCRIPTION
DATE	Timestamp	Timestamp for daily data
RAINFALL	Numerical	Rainfall happened during a day
MIN TEMP	Numerical	The maximum temperature reached in a day
MAX TEMP	Numerical	The minimum temperature reached in a day
MEDTEMP	Numerical	Average temperature
WINDSPEED	Numerical	Windspeed on a certain day

III. METHODOLOGY

1. Gaussian Processes

The Gaussian Processes are worldly used method of supervised learning for regression and classification (Holman et al., 2014). Supervised learning construes a hypothesis function $h(x)$ based on the training dataset. The training dataset consists of N number of vectors which consist of an input vector $X = \{x_1, \dots . x_n\}$ and the corresponding target vector $Y = \{y_1, \dots , y_n\}$ to generate input and output pairs $\{(x_i\ y_i), i = 1 \dots N\}$ the supervised learning model uses the training data to learn which intern builds an approximate model *h.* to evaluate the model h we use the testing data which was not given to building the model to test the model and estimate predictions, $\hat{y}(x)$ which are then compared with the ground truth (i.e., actual data). There are several statistical methods to validate the predicted outputs in the case of regression, RMSE, R squared error and MSE that can be used to determine the efficiency of the model *h* (Holman et al., 2014).

When it comes to gaussian regression we need to consider the posterior predictive distribution. In general, a posterior predictive distribution can be expressed as

$$P(y|x\ D) = \int_h P(y|x\ , h)\ P(h|D)dw \quad (1)$$

Where x is the test point given data D; y is the label for the given data, which we don't know yet for the test point x. The equation (1) depicts for the given x what will be the probability of label y

Using Bayes' rule, we can modify the same equation and marginalize and give it a prior

$$\frac{P(D|h) * p(h)}{Z} \quad (2)$$

If we consider the prior $P(h)$ which is a gaussian on its own when we multiply with the $P(D|h)$ which is another gaussian which will intern give us a gaussian and when we marginalize it we end up getting gaussian distribution. Now when multiplying the equation (1) and (2) we end up getting a gaussian distribution which is $P(y|x\ D)$. ("Gaussian Process", 2022)

With that said formula for the gaussian processes can be represented as the probability of label y given data point x with the respective the dataset

$$P(y|x, D) \sim \mathcal{N}(\mu, k) \quad (3)$$

In the above equation (3), μ is the mean of the dataset and k is the covariant function, on determining the mean and the covariant matrix we can find the predictions and gaussian distribution of all the points in the data set. This brings us to the most important step of performing gaussian processes, determining the covariant matrix.

1.1 Covariant Metrix/ Kernel

If we consider covariates metrics for one training point and one test point

$$p\left(\begin{bmatrix} y_1 \\ y \\ y_n \end{bmatrix} [x_1 \quad x \quad x_2]\right) \sim \mathrm{N}(\mu, k) \tag{4}$$

where K can look something like this

$$\mathrm{K} = \sum = \begin{bmatrix} \delta_1^2 & 3 \\ 3 & \delta_2^2 \end{bmatrix} \tag{5}$$

The δ diagonal represents the variants and the opposite diagonals represented how correlated is one data point with the other. The above example shows a high correlation between two data points since they are positive numbers. If the diagonals are 0 it means the data points do not affect each other in a gaussian.

When building a covariant Metrix for a dataset the general form of the matrix can be represented as

$$\sum \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix} \tag{6}$$

Where the K represents the training data, K_*^T is the test points and K** Is the variants of the test points and k* represents the variation of the training data.

1.2 Exponential Sine Squared Kernel

The exponential kernel allows one to model functions which can repeat themselves precisely("Gaussian Process summer school", 2022). The major parameters are length scale *l* and periodicity *P*. the expression of the kernel can be given as below

$$k(x_i, x_j) exp\left(2sin^2\left(\frac{\pi d(x_i, x_j)/P}{l^2}\right)\right) \quad (7)$$

Upon changing the value of l and the periodicity P we can determine the shape of the kernel matrix. A simple representation of the kernel is shown in figure 1 below

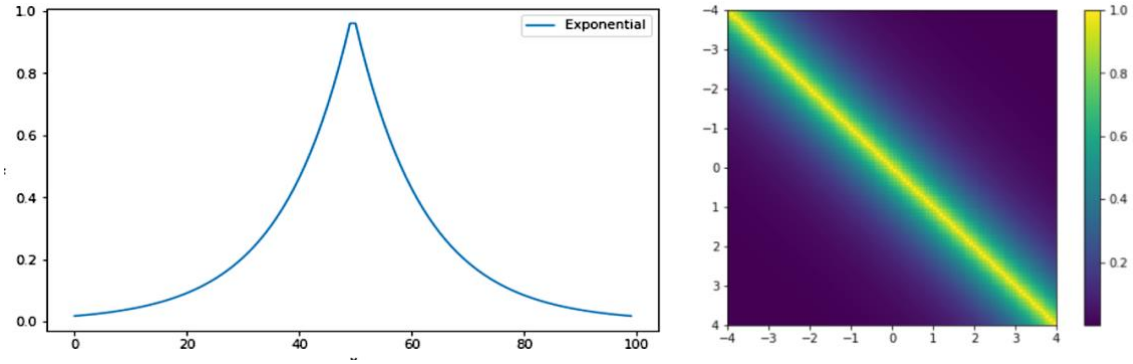


Figure 1: Exponential kernel plot

1.3 Radial Basis Function (RBF) Kernel

One of the most commonly used kernels in machine learning is the RBF kernel. The RBF kernel can be expressed as

$$K_{RBF}(x, x') = \sigma^2 exp\left(-\frac{(x - x')^2}{2l^2}\right) \quad (8)$$

The kernel has two main parameters which are variance σ^2 and lengthscale l. when there is some change in the two parameters we expect a change in the end output result. An example representation of the RBF kernel is shown in Figure 2 below.

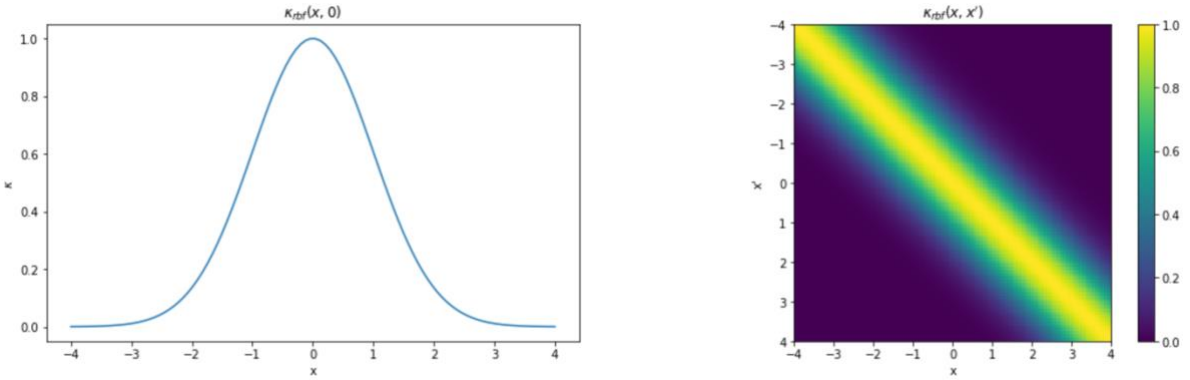


Figure 2: RBF kernel plot

It is quite evident that the variance and the lengthscale can control the shape of the output ("Gaussian Process summer school", 2022)
.

1.4 Multiplying Kernels

In a gaussian processes prediction depending on the results, we need we may need to add or multiply multiple kernels to get the required outputs this paper we multiply kernels to find the required outputs, and a plot of multiplying the RBF kernel and the exponential kernel is shown in figure 3. ("Gaussian Process summer school", 2022)

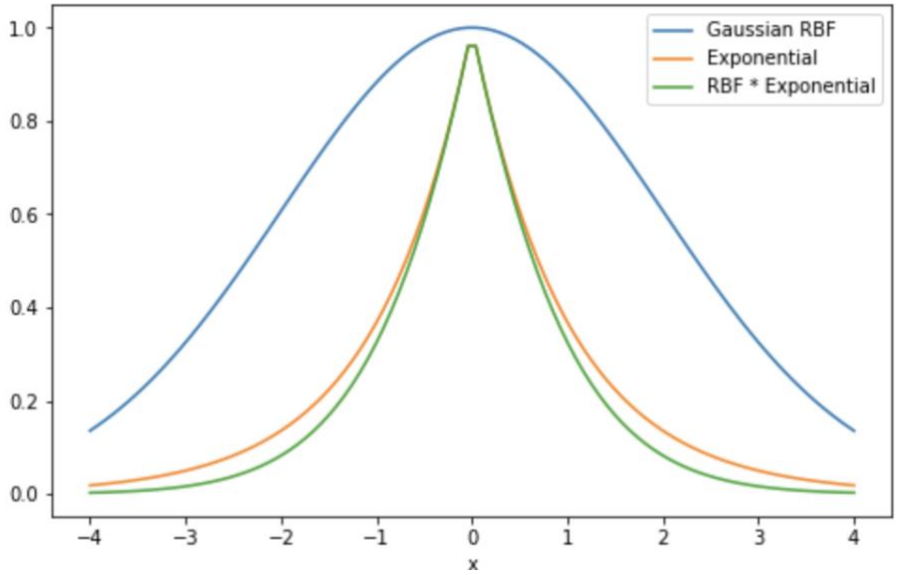


Figure 3:Product of

1.5 Predictions

Based on the above information on determining the mean of the dataset and the covariance matrix we can come up with predictions. The formula for producing predictions can be written as

$$P(y)(\, y_1 \ldots . y_n \,,\; x_1 \ldots . x_n) \sim \mathrm{N}(k_*^T\, k_y^{-1}, k_{**} \,-\, k^{-1} k_y) \tag{9}$$

Therefore for predictions, we can use the posterior mean and then use the predictive variants for measuring the confidence intervals ("Gaussian Process", 2022).

IV. EXPERIMENTAL SETUP

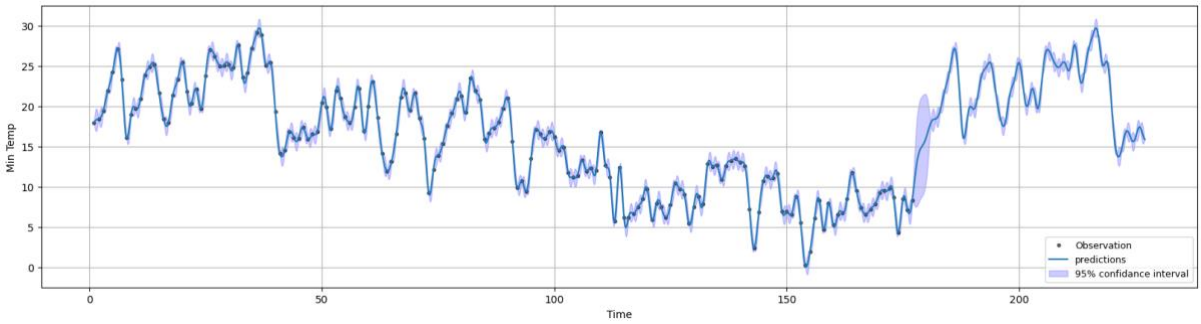
The experiments performed in this project were implemented in google collab an open environment where python can be executed, the libraries used in this project were NumPy for array creation and selection Pandas for importing the data frame and Sklearn libraries were imported for performing machine learning and to visualise the outputs matplotlib was utilised. GPy libraries were also made use when describing the kernel function. The necessary variables were extracted from the dataset which was our training set and a test set was created for predicting 177 parameters. The data was subjected to some anomalies which were addressed by removing the null values and also dropping duplicate values and finally, the data were normalised. Upon performing the necessary data pre-processing steps we fit the data to make predictions and validate the model with the testing set. Finally with the use of regression metrics which were imported from the Sklearn library the models were validated.

V. EXPERIMENTAL RESULTS

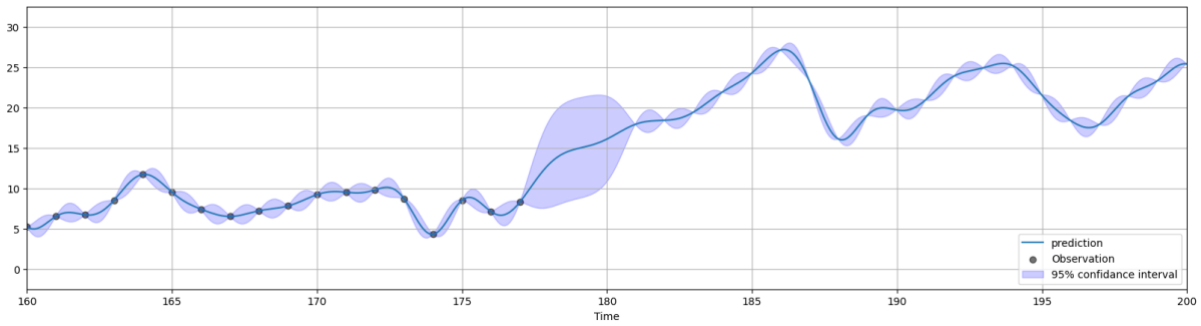
1. Exponential Sign Square Model

In Figure 4 model output indicates excellent predictions over the observer point, the model was able to find the uncertainties for each

point and give us an estimate, this can intern give us an inference on how much the minimum temperature can be expected in a particular day, the model was also able to predict 50 days in the future and also give uncertainties. With this knowledge, we can pre-plan our future events like crop plantation or conducting an event on a large scale. The model was used to implement the same with the max temperature, rainfall, humidity and wind speed. A domain expert can use this information to make decisions for the future. Model implementation outputs on max temperature, humidity, rainfall, and windspeed can be found in the appendix-A section



(A) GP minimum temperature estimate using Exponential sign Square kernel

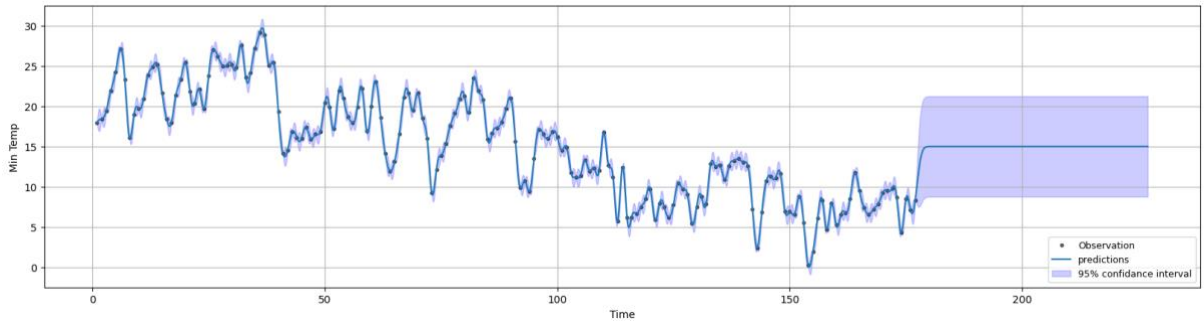


(B) GP future minimum temperature uncertainty estimate

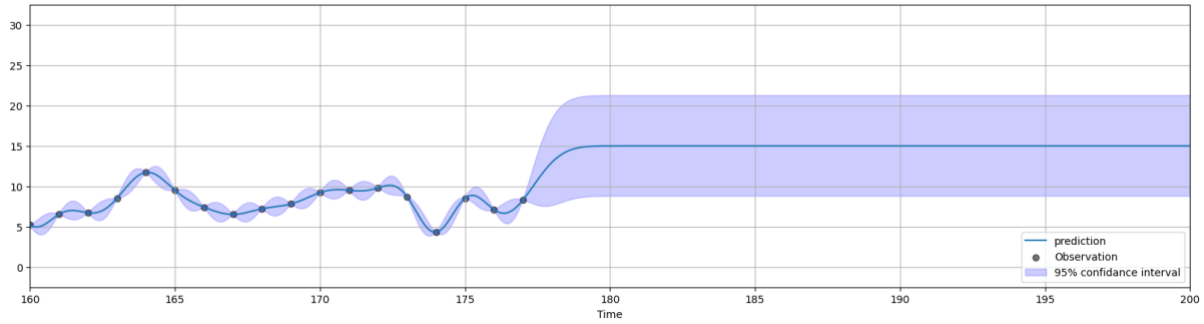
Figure 4: GP model prediction for Minimum temperature in Cobar

2. Radial Basis Function Model

The below figure(5) depicts the outputs for the RBF model. The model was able to clearly predict the given data points and also find the uncertainties but the model failed to find predictions when there are no data points as shown in the image this may have some similarities with the RNN's vanishing gradient problem. The model was also deployed with max temperature, humidity, rainfall, and windspeed and the outputs can be found in the appendix-B section.



(A) GP minimum temperature estimate using RBF model

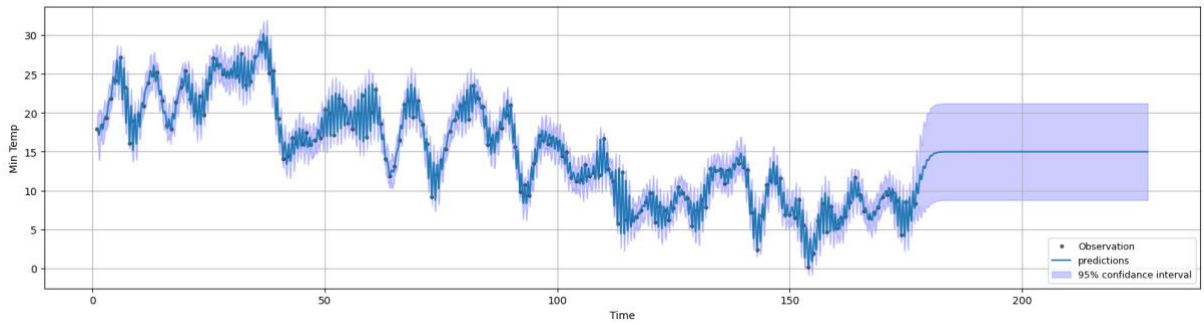


(B) GP future minimum temperature uncertainty estimate

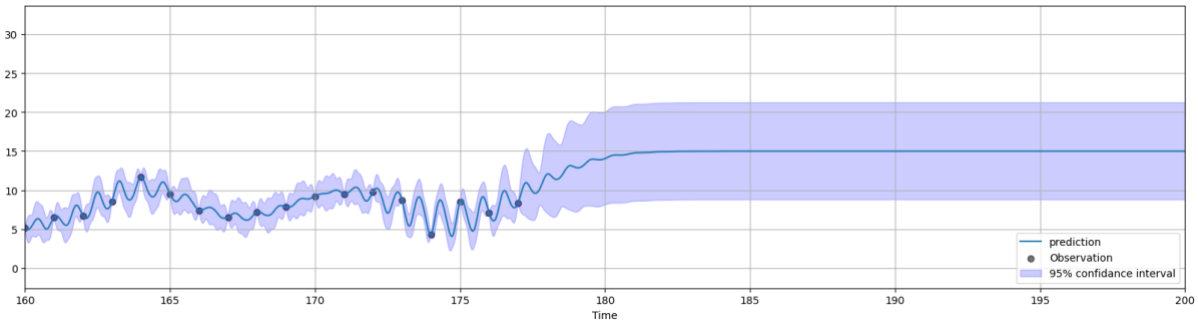
Figure 5: GP model prediction for Minimum temperature with RBF model

3. Combination Model

In this model, we intend to multiply the above two kernels (RBF and Exponential Sign Square) and determine the model predictions. The model outputs are shown in figure (6). It was expected for this model to perform well, on multiplying two kernels but on the contrary model has performed poorly than the other two models above in figure (6)(A) the prediction was not good enough and in figure (B) we can see the uncertainties for each point is not properly defined and when predicting in the future it failed to give prediction like the RBF model.



(A) GP minimum temperature estimate using RBF model



(B) GP future minimum temperature uncertainty estimate

Figure 6 GP model prediction with a combination model

Cobar station	Exponential Sign Square Model			RBF Model			Combination Model		
	R^2	MSE	RMSE	R^2	MSE	RMSE	R^2	MSE	RMSE
Min Temp	-1.05	87.59	9.359	-1.05	87.59	9.359	-1.25	96.18	9.80
Max Temp	-1.87	201.1	14.18	-1.87	201.1	14.18	-1.73	191.2	13.82
Rainfall	-0.09	135.5	11.643	-0.09	135.5	11.643	-0.03	128.6	11.34
Wind Speed	-0.08	271.4	5.209	-0.08	271.4	5.209	-0.07	27.08	5.204
Humidity	-1.13	903.3	30.05	-1.13	903.3	30.05	-1.09	884.3	29.73

Table 2 Performance measure for estimates obtained from GP models at Cobar station from 1-1-2009 till 7-5-2009. The Exponential model and the RBF predeceases similar model accuracy.

Table 2 summarises the performance of the GP models in one particular station, using the performance measures (R^2 , MSE and RMSE) we can say that model one was better at prediction and it also excelled in finding the uncertainties when no data was given and it was able to give good prediction 50 days ahead in the future. On the other hand model, 2 gave us a prediction but the model wasn't tuned enough to give a future prediction. On contrary, model 3 gave us the least accuracy and predictions made no sense therefore no inference can be taken from it. NOTE: All the models used were Non- linear models

VI. DISCUSSION

For this paper we have produced some predictions for the given weather data using gaussian processes regression models, the regression can be performed with a simple RNN or a support vector machine but we cannot find uncertainties for a given datapoint and may have vanishing gradient issue to mitigate this problem the natural next step or an alternative approach for a regression problem is using GP. ("Gaussian Processes for Dummies ·", 2022)

The use of different kernels gave us a better idea of how can a kernel function affect the results it turns out the most commonly used kernel "RBF" didn't give us the results which was expected on the other hand the desired results were produced by the model 1 which used the exponential sign square kernel and the final model which was a combination of both the kernels didn't give us any information and performed poorly when compared.

Although the RBF model and the combination model didn't give the expected performance, there is always room for improvement, by tuning the hyperparameters like the l value or the σ value we may end up with a different result all to gather.

The huge uncertainty shown in the model 1 outputs when predicted 50 days indicates when there is a large gap between two data point the uncertainties seem higher which made sense. It was also found that the model was outputting the same values when e try predicting 50 days. Which was very similar to the starting 0-50 points and the endpoints where there was no data point. This was subjected to a mystery on why this phenomenon took place and further study needs to be done on this part.

Given its reliance on food production and gold mining, the town of Cobar can greatly benefit from this information. The aforementioned forecast and uncertainty can provide them with an estimate of the weather for the upcoming days, which can assist farmers and miners in

planning their tasks based on the local weather. By adhering to the prediction and the uncertainty, a miner and a farmer may safeguard their crops and the equipment they use for mining and farming against natural disasters. As a result, the model prediction could spare the employees and owners from suffering a substantial loss.

When compared with the state-of-the-art papers in regards to gaussian processes. Many authors have used deep gaussian processes and also GP for huge datasets which is a separate topic on its own. Though our paper pales on compression future studies can be performed with deep GPs and also perform GP for huge datasets and find better connivance intervals between each point.

There were absolutely no legal and ethical constraints on conducting these experiments and the dataset used in this paper originated from the national bureau of meteorology Australia the data is open-sourced and can be used for performing any tasks which it is beneficial to society as a whole. No person or animal was harmed while conducting this experiment.

VII. CONCLUSION

As a result of our use of GP to do regression on a specific place in Australia, it has been discovered that this programme is a very potent tool with many applications, including speech recognition and self-driving automobile vehicle. The above issue can also be resolved by combining several kernels and fine-tuning the hyperparameters. Since there are numerous elements influencing climate change in particular, including PCA and feature selection can assist us in identifying the variables that contribute the most to a large dataset.

VII. REFERENCES

Chen, N., Qian, Z., Nabney, I., & Meng, X. (2014). Wind Power Forecasts Using Gaussian Processes and Numerical Weather Prediction. *IEEE Transactions On Power Systems*, *29*(2), 656-665. doi: 10.1109/tpwrs.2013.2282366

Holman, D., Sridharan, M., Gowda, P., Porter, D., Marek, T., Howell, T., & Moorhead, J. (2014). Gaussian process models for reference ET estimation from alternative meteorological data sources. *Journal Of Hydrology*, *517*, 28-35. doi: 10.1016/j.jhydrol.2014.05.001

Gaussian Process. (2022). Retrieved 10 August 2022, from https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html

Gaussian Process summer school. (2022). Retrieved 11 August 2022, from http://gpss.cc/gpss19/

Görtler, J., Kehlbeck, R., & Deussen, O. (2022). A Visual Exploration of Gaussian Processes. Retrieved 12 August 2022, from

Gaussian Processes for Dummies ·. (2022). Retrieved 12 August 2022, from http://katbailey.github.io/post/gaussian-processes-for-dummies/
Kalnay, E., Kanamitsu, M., & Baker, W. (1990). Global Numerical Weather Prediction at the National Meteorological Center. *Bulletin Of The American Meteorological Society*, *71*(10), 1410-1428. doi: 10.1175/1520-0477(1990)071<1410:gnwpat>2.0.co;2

Seo, K., Wang, W., Gottschalck, J., Zhang, Q., Schemm, J., Higgins, W., & Kumar, A. (2009). Evaluation of MJO Forecast Skill from Several Statistical and Dynamical Forecast Models. *Journal Of Climate*, *22*(9), 2372-2388. doi: 10.1175/2008jcli2421.1

Climate Change: Global Temperature, C. (2022). Climate Change: Global Temperature. Retrieved 14 August 2022, from https://www.climate.gov/news-features/understanding-climate/climate-change-global-temperature

Appendix A – Importing libraries and performing model 1

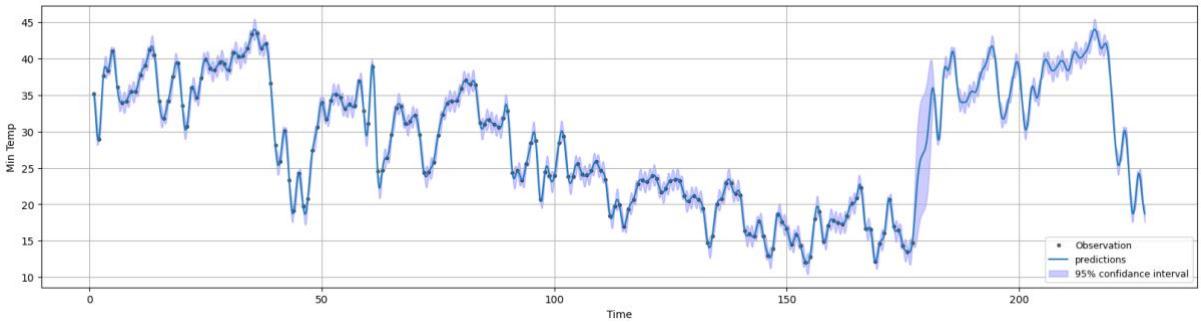
Dataset Link – https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package

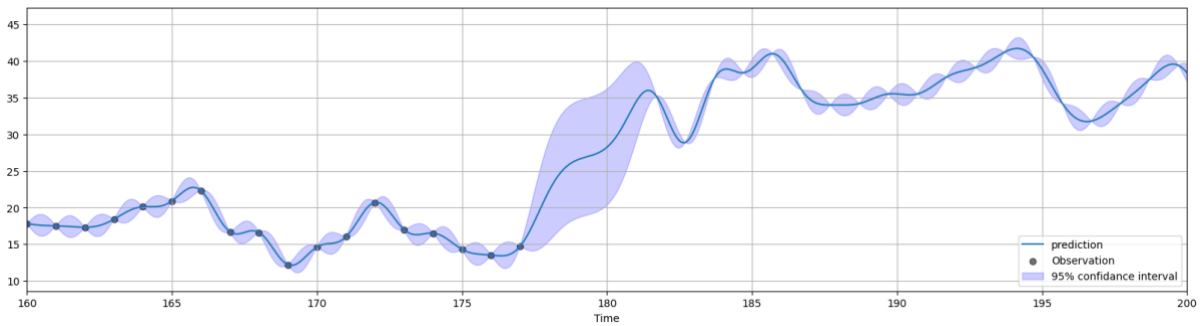
```
import NumPy as np
import pandas as PD
import matplotlib.pyplot as plt

from scipy import linalg
from sklearn import metrics
from sklearn import linear_model
from sklearn import preprocessing
import seaborn as sns
import scipy

from sklearn.gaussian_process.kernels import ConstantKernel as C, RBF, RationalQuadratic
as RQ, WhiteKernel, ExpSineSquared as Exp , DotProduct as DP , ExpSineSquared , Matern,
RBF,RationalQuadratic
from sklearn.gaussian_process.kernels import Hyperparameter
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, Matern, DotProduct, WhiteKernel,
RationalQuadratic, ExpSineSquared


df = pd.read_csv("/content/weatherAUS.csv")
df.dropna(inplace=True)
df.head()



df_array = np.asarray(df)

date = df_array[0:177,0]
MinTemp = df_array[0:177,2]
MaxTemp = df_array[0:177,3]
Rainfall = df_array[0:177,4]
```

```
windspeed = df_array[0:177,8]
humidity = df_array[0:177,13]

y = np.asarray([MinTemp,MaxTemp,windspeed,Rainfall,humidity]).T

X = np.atleast_2d([1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42 ,43 ,44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54 ,55, 56, 57 ,58 ,59, 60, 61, 62, 63 ,64, 65, 66 ,67, 68, 69, 70 ,71, 72, 73,
74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85 ,86, 87 ,88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120 ,121, 122, 123, 124, 125, 126, 127, 128 ,129, 130, 131, 132, 133, 134, 135,
136, 137 ,138 ,139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153,
154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165 ,166 ,167, 168, 169 ,170, 171,
172, 173 ,174, 175, 176, 177]).T


x= np.atleast_2d(np.linspace(X.min(),X.max()+50,10739)).T #10739

kernel = C()*Exp(length_scale=24,periodicity= 1)

gp  =  GaussianProcessRegressor(kernel=kernel,  normalize_y=True,  n_restarts_optimizer=
400)
gp.fit(X,y)

print("\nLearned kernel:", gp.kernel_)
print("Log-marginal-likelihood: %.3f" % gp.log_marginal_likelihood(gp.kernel_.theta))

#prediction
y_pred_1, sigma_1 = gp.predict(x,return_std=True)

#plotting model _1
fig = plt.figure( figsize=(20,5), dpi = 100, facecolor="w", edgecolor = "k")
```

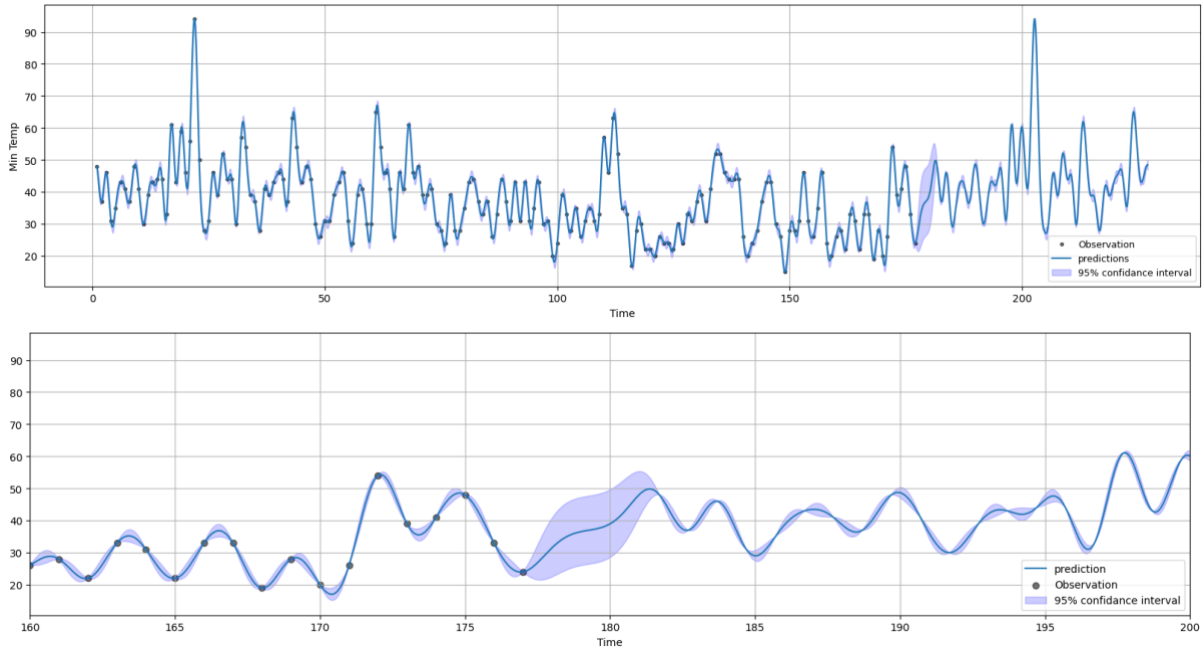
```
#plt.subplot(4,1,1)
plt.plot(X,y[:,0],"k.",alpha=0.55, label ='Observation')
plt.plot(x,y_pred_1[:,0],label ='predictions')
#plt.plot([min(y_pred_1),max(y_pred_1)], [min(y_pred_1),max(y_pred_1)], ls="--", c=".3")
plt.fill_between(x[:,0], y_pred_1[:,0] - sigma_1[:,0], y_pred_1[:,0] + sigma_1[:,0], alpha = 0.2,
color = "blue", label = "95% confidance interval")
plt.xlabel("Time")
plt.ylabel("Min Temp")
plt.legend(loc="lower right", fontsize = 9)
plt.grid()
#plt.xlim(60,70)
#plt.ylim(0,50)
```

Plots for Max Temperature Model 1

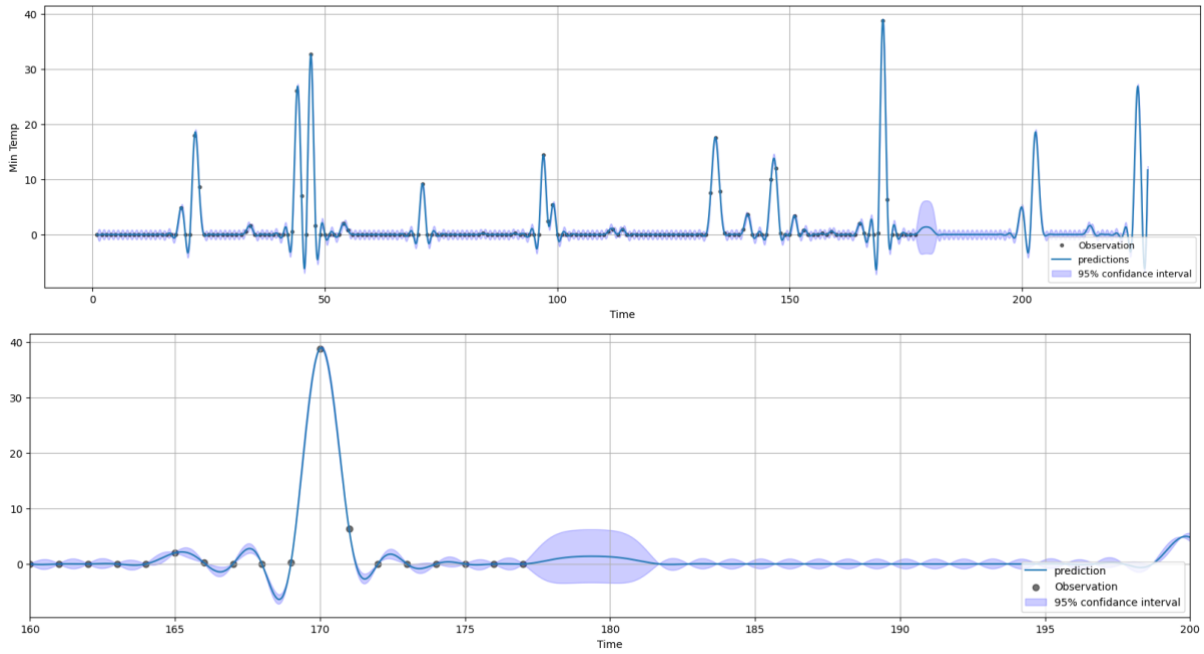




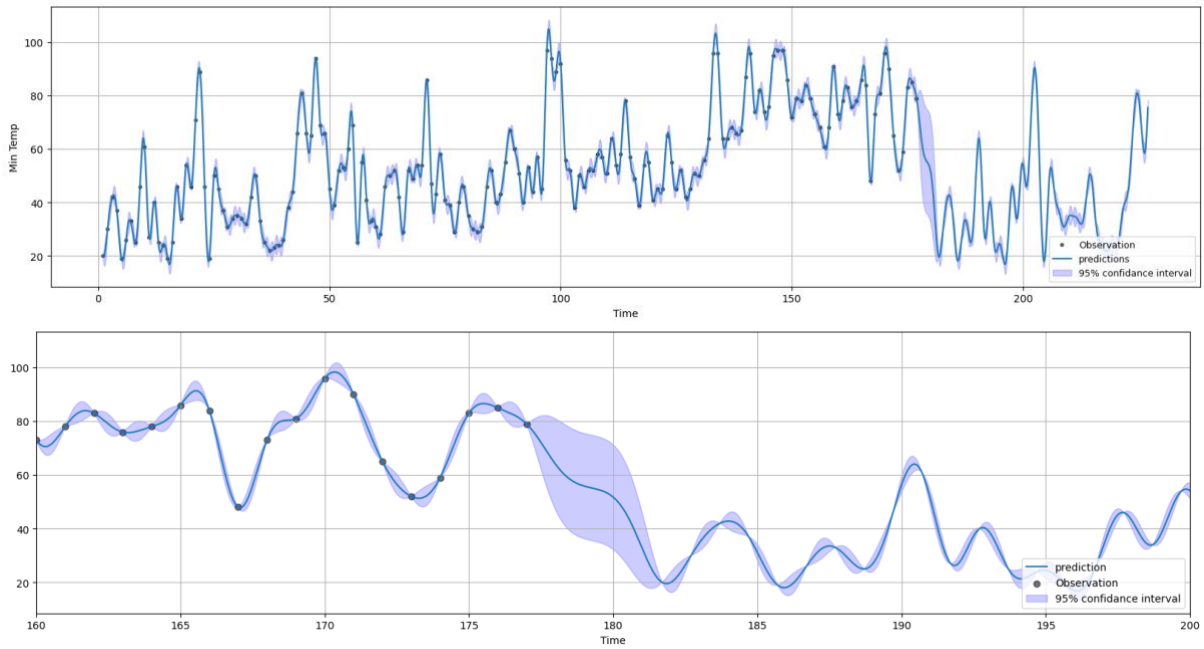
Plots for Rainfall Model 1



Plot for windspeed Model 1



Plot for humidity Model 1

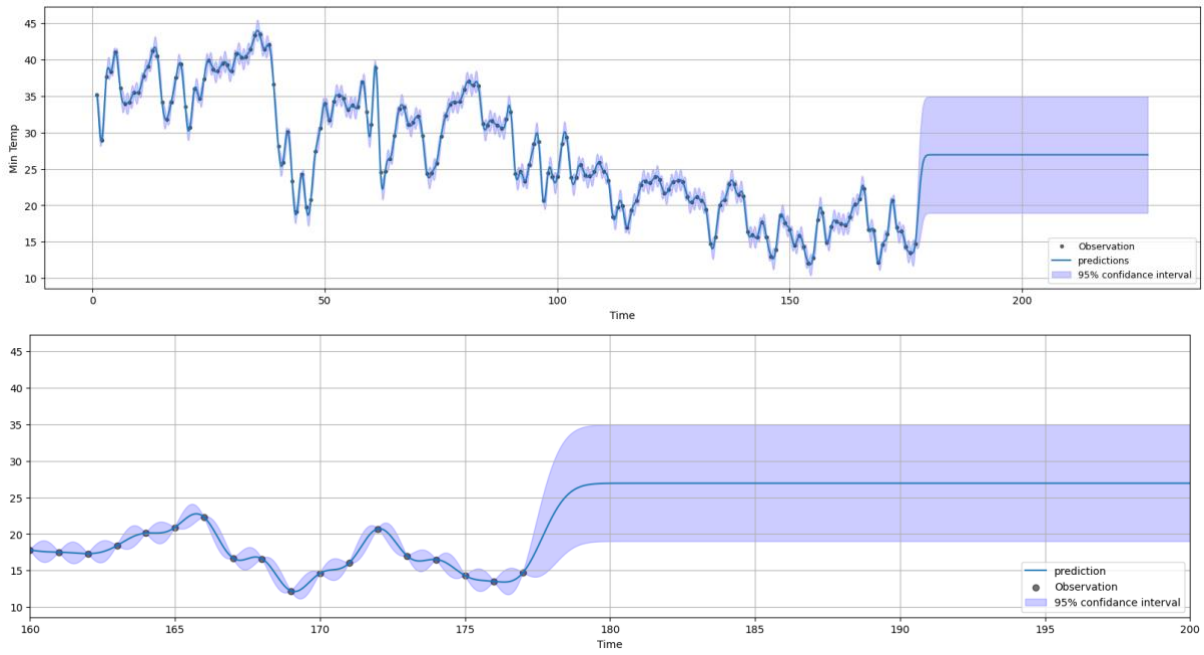



```
from math import sqrt
#print(y_pred_1[0:177:,0])
#MinTemp_pred = df_array[177:195,2]
#print((MinTemp_pred))
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_log_error
print(r2_score(y[:,1],y_pred_2[0:177:,1]))
#print(mean_absolute_error(y[:,0],y_pred_1[0:177:,0]))
print(mean_squared_error(y[:,1],y_pred_2[0:177:,1]))
#print(mean_squared_log_error(y[:,0],y_pred_1[0:177:,0]))
print(sqrt(mean_squared_error(y[:,1],y_pred_2[0:177:,1])))
```

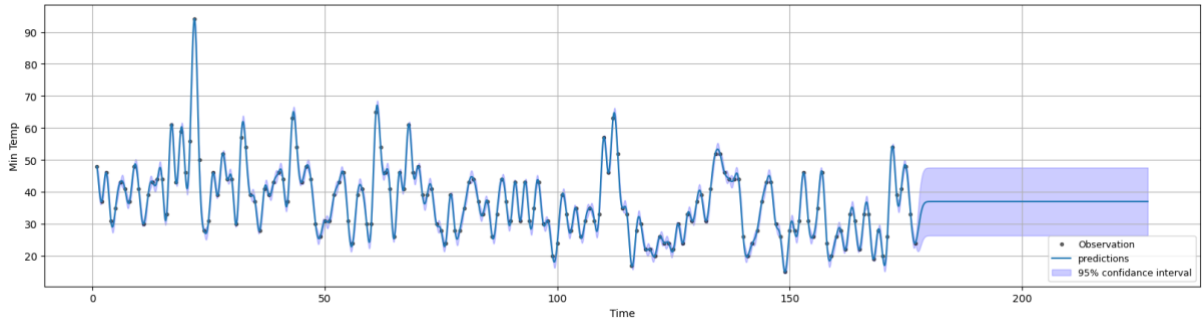
Appendix B – Plots for model 2

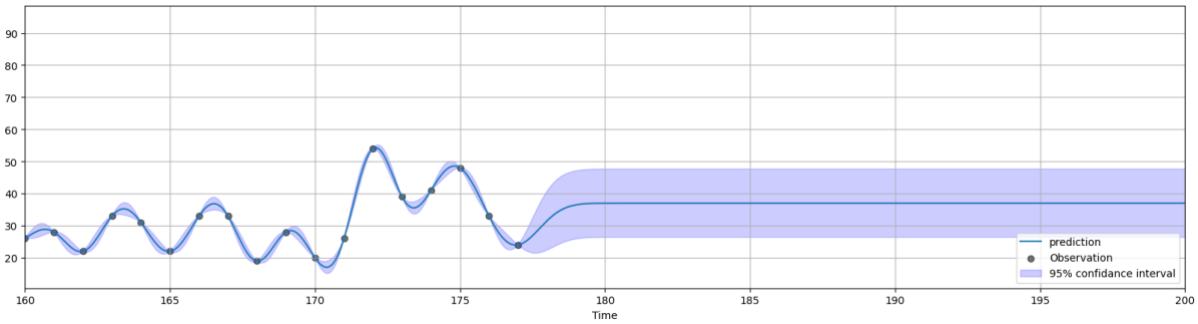
```
#model 2
kernel_2 = 1.0 * RBF(length_scale = 24)
#gp = GaussianProcessRegressor(kernel=kernel_2, normalize_y=True, n_restarts_optimizer=
400)
#gp.fit(X,y)

print("\nLearned kernel:", gp.kernel_)
print("Log-marginal-likelihood: %.3f" % gp.log_marginal_likelihood(gp.kernel_.theta))

y_pred_2, sigma_2 = gp.predict(x,return_std=True)

#plotting model _1
fig = plt.figure( figsize=(20,5), dpi = 100, facecolor="w", edgecolor = "k")

#plt.subplot(4,1,1)
plt.plot(X,y[:,0],"k.",alpha=0.55, label ='Observation')
plt.plot(x,y_pred_2[:,0],label ='predictions')
#plt.plot([min(y_pred_1),max(y_pred_1)], [min(y_pred_1),max(y_pred_1)], ls="--", c=".3")
plt.fill_between(x[:,0], y_pred_2[:,0] - sigma_2[:,0], y_pred_2[:,0] + sigma_2[:,0], alpha = 0.2,
color = "blue", label = "95% confidance interval")
plt.xlabel("Time")
plt.ylabel("Min Temp")
plt.legend(loc="lower right", fontsize = 9)
plt.grid()
#plt.xlim(60,70)
#plt.ylim(0,50)


# Plot
fig = plt.figure(figsize=(20,5), dpi = 100, facecolor="w", edgecolor = "k")
```

```
plt.scatter(X, y[:,0], c='k', alpha=0.55, label ='Observation')
plt.plot(x,y_pred_2[:,0], label ='prediction')
plt.fill_between(x[:,0], y_pred_2[:,0] - sigma_2[:,0], y_pred_2[:,0] + sigma_2[:,0], alpha = 0.2,
color = "blue", label = "95% confidance interval")
plt.xlim(160, 200)
plt.xlabel("Time")
plt.legend(loc="lower right", fontsize = 10)
plt.grid()
```

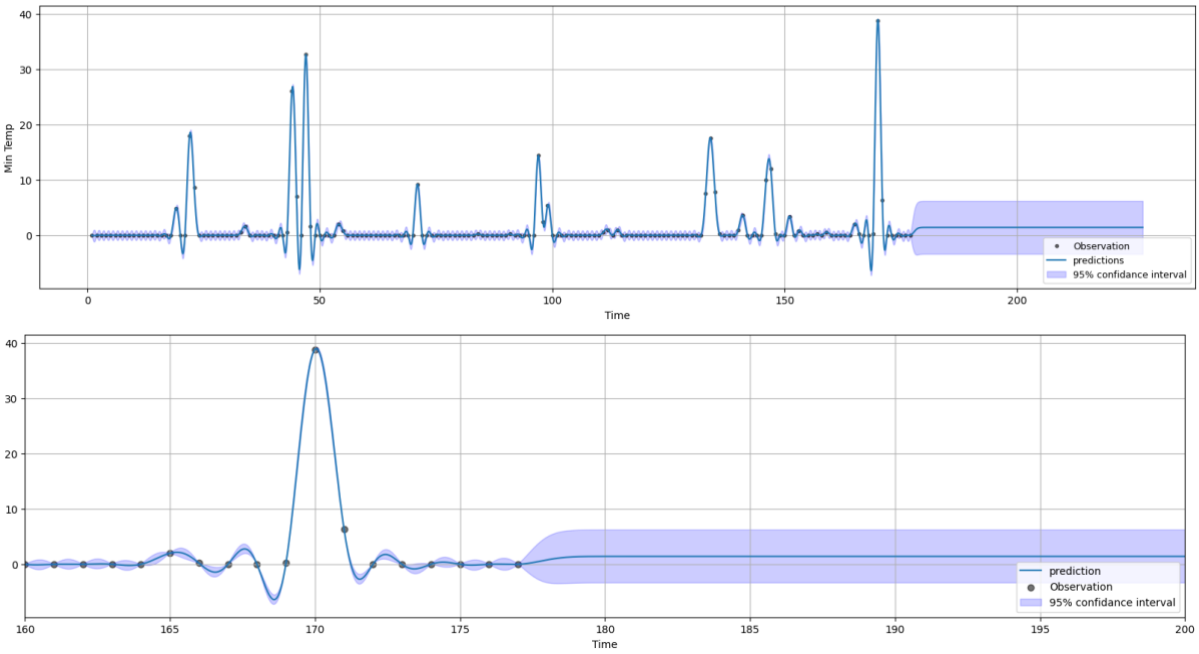


Plots for Rainfall model 2

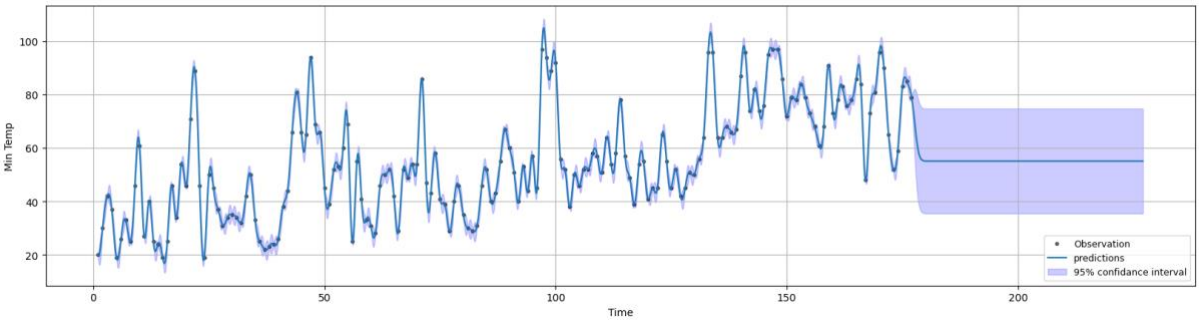


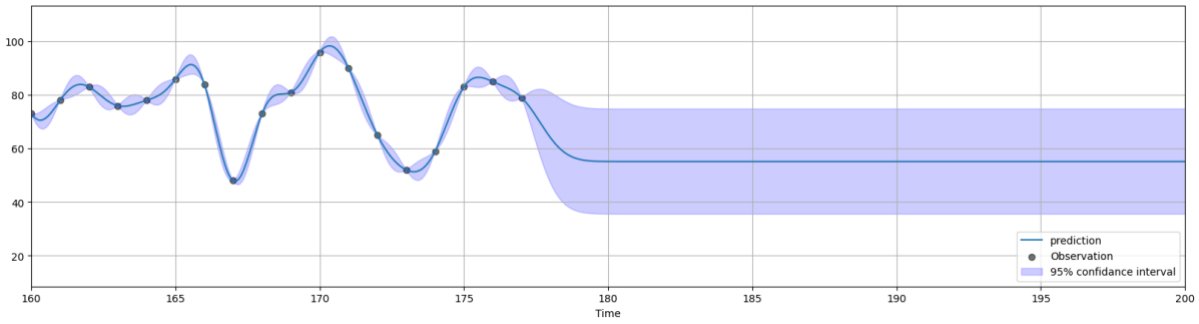


Plots for windspeed model 2



Plots for humidity model 2






```
from math import sqrt
#print(y_pred_1[0:177:,0])
#MinTemp_pred = df_array[177:195,2]
#print((MinTemp_pred))
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_log_error
print(r2_score(y[:,1],y_pred_2[0:177:,1]))
#print(mean_absolute_error(y[:,0],y_pred_1[0:177:,0]))
print(mean_squared_error(y[:,1],y_pred_2[0:177:,1]))
#print(mean_squared_log_error(y[:,0],y_pred_1[0:177:,0]))
print(sqrt(mean_squared_error(y[:,1],y_pred_2[0:177:,1])))
```

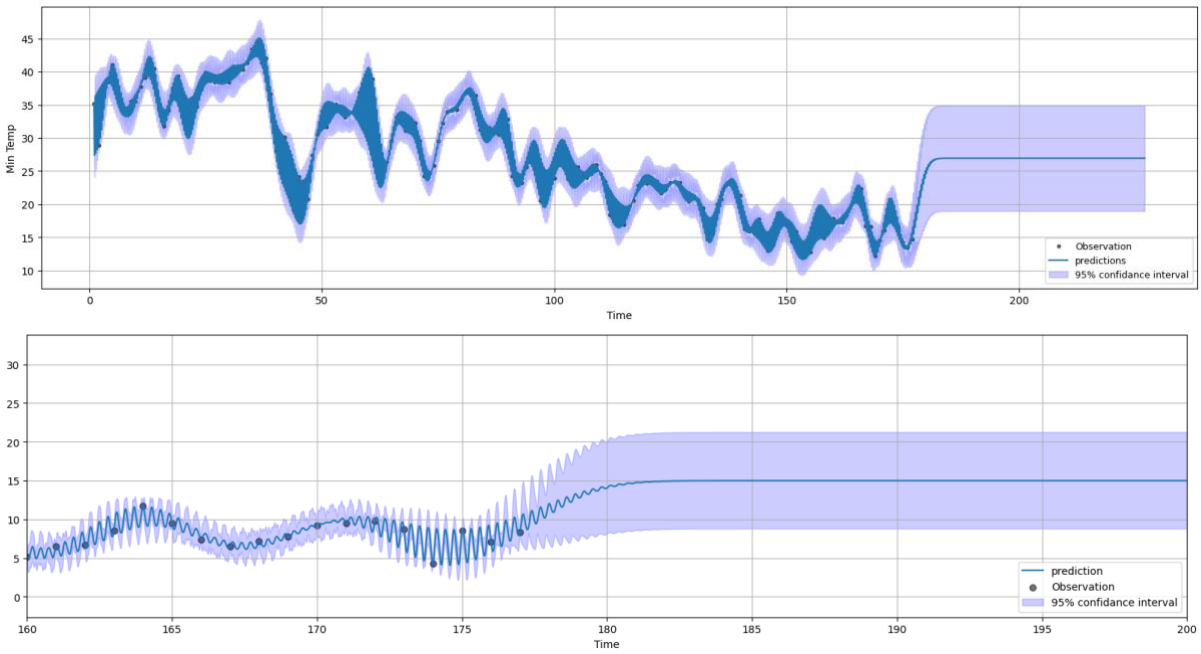
Appendix – C plots for model 3

```
#model_3
```

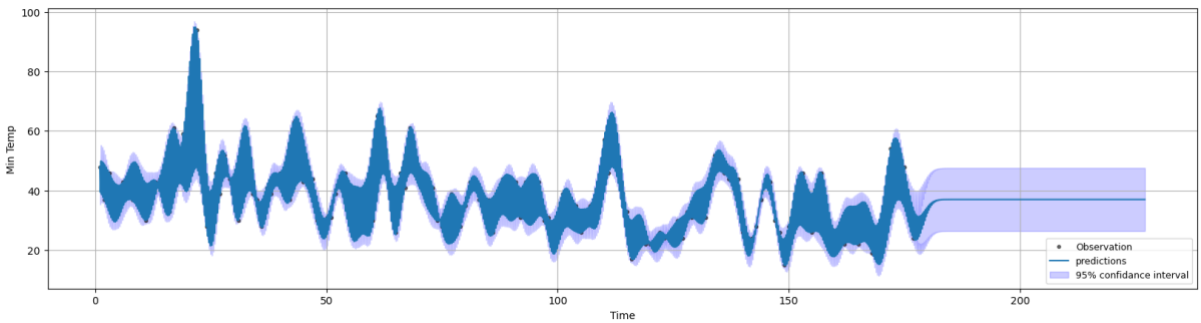
```
kernel_3 = kernel_2 * kernel
gp = GaussianProcessRegressor(kernel=kernel_3, normalize_y=True, n_restarts_optimizer=
400)
gp.fit(X,y)
print("\nLearned kernel:", gp.kernel_)
print("Log-marginal-likelihood: %.3f" % gp.log_marginal_likelihood(gp.kernel_.theta))


y_pred_3, sigma_3 = gp.predict(x,return_std=True)

#plotting model _1
fig = plt.figure( figsize=(20,5), dpi = 100, facecolor="w", edgecolor = "k")

#plt.subplot(4,1,1)
plt.plot(X,y[:,1],"k.",alpha=0.55, label ='Observation')
plt.plot(x,y_pred_3[:,1],label ='predictions')
#plt.plot([min(y_pred_1),max(y_pred_1)], [min(y_pred_1),max(y_pred_1)], ls="--", c=".3")
plt.fill_between(x[:,0], y_pred_3[:,1] - sigma_3[:,1], y_pred_3[:,1] + sigma_3[:,1], alpha = 0.2,
color = "blue", label = "95% confidance interval")
plt.xlabel("Time")
plt.ylabel("Min Temp")
plt.legend(loc="lower right", fontsize = 9)
plt.grid()
#plt.xlim(60,70)
#plt.ylim(0,50)

# Plot
fig = plt.figure(figsize=(20,5), dpi = 100, facecolor="w", edgecolor = "k")
plt.scatter(X, y[:,0], c='k', alpha=0.55, label ='Observation')
plt.plot(x,y_pred_3[:,0], label ='prediction')
plt.fill_between(x[:,0], y_pred_3[:,0] - sigma_3[:,0], y_pred_3[:,0] + sigma_3[:,0], alpha = 0.2,
color = "blue", label = "95% confidance interval")
plt.xlim(160, 200)
```

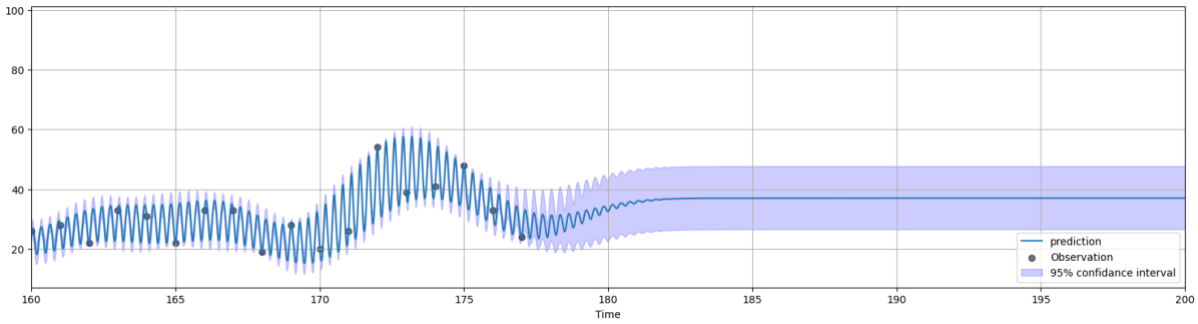
```
plt.xlabel("Time")
plt.legend(loc="lower right", fontsize = 10)
plt.grid()
```

plot for max temperature model 3

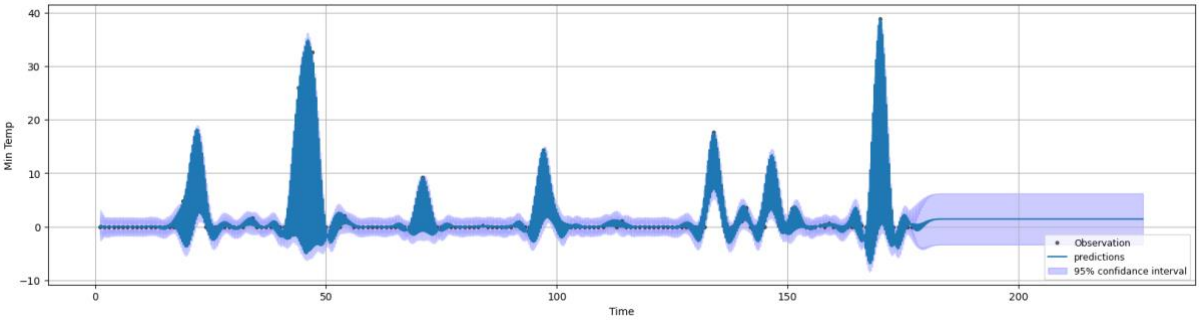


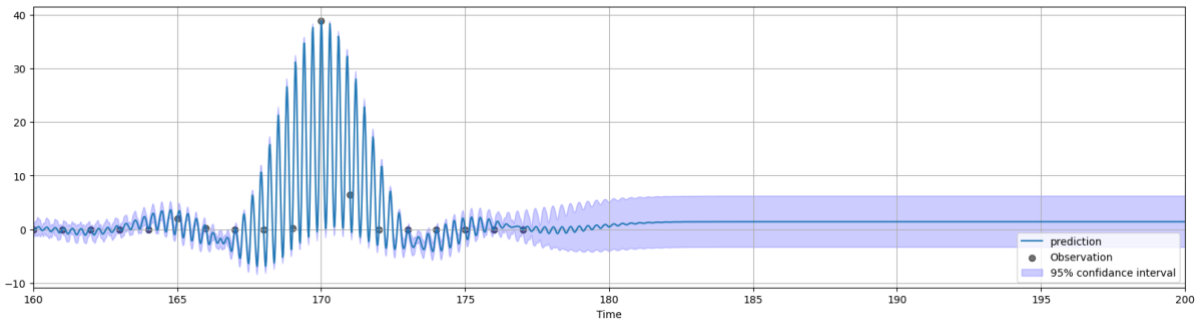
plot for rainfall model 3



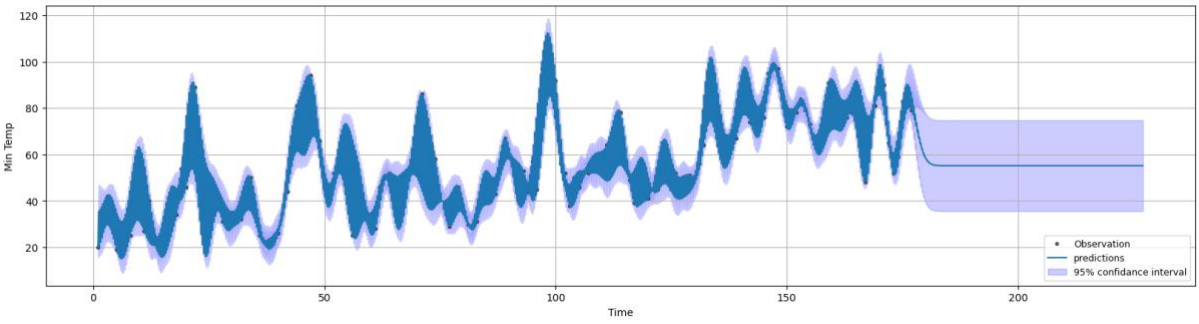


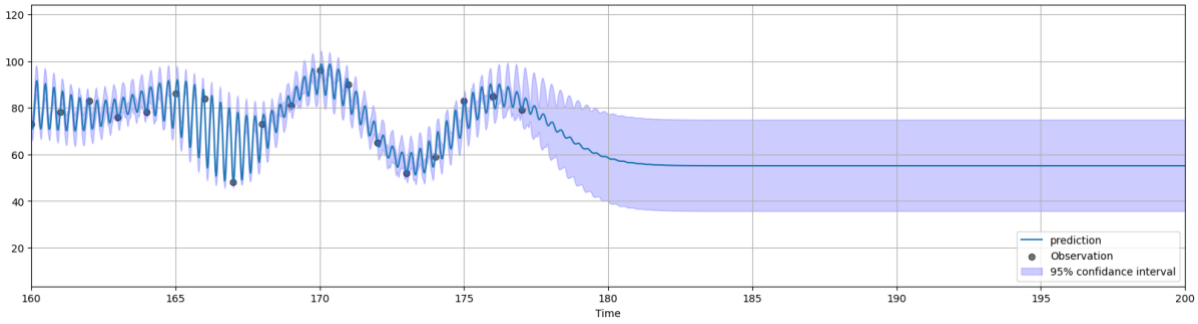
plot for humidity model 3





plot for windspeed model 3






```
from math import sqrt
#print(y_pred_1[0:177:,0])
#MinTemp_pred = df_array[177:195,2]
#print((MinTemp_pred))
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_log_error
print(r2_score(y[:,4],y_pred_3[0:177:,4]))
#print(mean_absolute_error(y[:,0],y_pred_1[0:177:,0]))
print(mean_squared_error(y[:,4],y_pred_3[0:177:,4]))
#print(mean_squared_log_error(y[:,0],y_pred_1[0:177:,0]))
print(sqrt(mean_squared_error(y[:,4],y_pred_3[0:177:,4])))
```

Appendix D – Kernel plots with Gpy

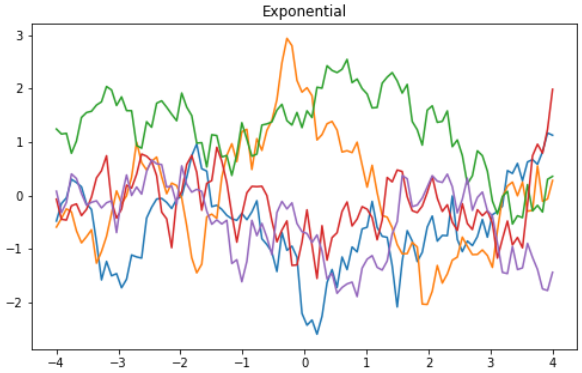
```
!pip install Gpy
```

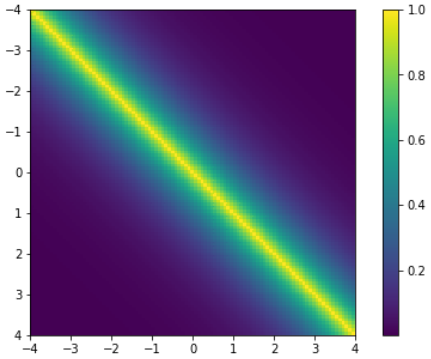
```
# Support for maths
import numpy as np
# Plotting tools
from matplotlib import pyplot as plt
# we use the following for plotting figures in jupyter
```

```
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

# GPy: Gaussian processes library
import GPy

# Create a 1-D RBF kernel with default parameters
k = GPy.kern.RBF(1)
# Preview the kernel's parameters
k

# Our sample space: 100 samples in the interval [-4,4]
X = np.linspace(-4.,4.,100)[:, None] # we need [:, None] to reshape X into a column vector for use in GPy

# Set up the plotting environment
plt.figure(figsize=(18,5))

# ==== k(x,0)

plt.subplot(121) # left plot

# First, sample kernel at x' = 0
K = k.K(X, np.array([[0.]])) # k(x,0)

# Plot covariance vector
plt.plot(X,K)

# Annotate plot
plt.xlabel("x"), plt.ylabel("$\kappa$")
```

```
plt.title("$\kappa_{RBF}(x,0)$")

# ==== k(x,x')

plt.subplot(122) # right plot

# The kernel takes two inputs, and outputs the covariance between each respective point in the two inputs
K = k.K(X,X)

# Plot the covariance of the sample space
plt.pcolor(X.T, X, K)

# Format and annotate plot
plt.gca().invert_yaxis(), plt.gca().axis("image")
plt.xlabel("x"), plt.ylabel("x'"), plt.colorbar()
plt.title("$\kappa_{RBF}(x,x')$");

ks = [
    GPy.kern.Exponential(1),

]
# The name of our kernels (for the legend)
kernel_name = ["Exponential", "RBF ls=0.5", "RBF ls=0.25, var=2", "Exponential", "Matern 3/2",
               "Matern 5/2", "Periodic", "Cosine", "Brownian", "Linear", "Bias", "White", "Periodic x RBF", "Linear + Exponential"]

# Our sample space
X = np.linspace(-4., 4., 100)[:, None]
```

```
print("The following plots demonstrate samples from a Gaussian process prior and the corresponding covariance matrix")

# Loop through our kernels
for i,k in enumerate(ks):
    # The mean function is set to 0
    mu = np.zeros((100)) # we have 250 sample inputs
    # Get the covariance matrix
    if i is not 11:
        C  = k.K(X,X)
    else: # We have to sample White noise kernel differently
        C = k.K(X)

    # Sample 5 times from a multivariate Gaussian distribution with mean 0 and covariance k(X,X)
    Z  = np.random.multivariate_normal(mu, C, 5)

    # Setup figure environment
    plt.figure(figsize=(18, 5))

    # Show samples on left hand side
    plt.subplot(121)
    for j in range(5 if i < 11 else 2): # Loop through samples
        plt.plot(X[:],Z[j,:])
        plt.title(kernel_name[i])

    # Visualise covariance matrix on right hand side
    plt.subplot(122)
    plt.pcolor(X.T, X, C)
    # Annotate plot
    plt.gca().invert_yaxis(), plt.gca().axis("image")
    plt.colorbar()
```





```
# Create the first kernel: a 1-D RBF with lengthscale 2.0
k_R = GPy.kern.RBF(1, lengthscale=2., name="RBF")
# Create the second kernel: a 1-D Matern52 with variance 2.0 and lengthscale 4.0
k_E = GPy.kern.Exponential(1)#(1, variance=2., lengthscale=4., name="Matern52")

# Add the kernels together
k_sum = k_R * k_E
# Preview the properties of the composite kernel
k_sum

# Our sample space : 100 samples in the interval [-10,10]
X = np.linspace(-4., 4., 100)[:,None]

# Set up the plotting environment
plt.figure(figsize=(8,5))

# Here we sample from the consituent and composite kernels
K_R   = k_R.K(X, np.array([[0.]]))    # RBF
K_E   = k_E.K(X, np.array([[0.]]))
K_sum = k_sum.K(X, np.array([[0.]]))  # RBF + expo
```

```
# Plot each of our covariance vectors
plt.plot(X, K_R, X, K_E, X, K_sum)
#plt.plot(K_E)
# Annotate plot
plt.xlabel("x"), plt.ylabel("$\kappa$")
plt.legend(labels=["Gaussian RBF", "Exponential", "RBF * Exponential"]);
#plt.legend(labels=[ "Exponential"]);
```

Appendix E – Other kernel outputs

```
# -*- coding: utf-8 -*-
"""weather data gp.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/19nTN0AIxyJyQEqXXr3UZQvZ0cTMZzCCc
"""

import pandas as pd
import numpy as np
import sklearn.gaussian_process as gp
from sklearn.gaussian_process.kernels import ConstantKernel as C, RBF, RationalQuadratic as RQ, WhiteKernel , ExpSineSquared as Exp , DotProduct as DP , ExpSineSquared , Matern, RBF,RationalQuadratic
from sklearn.gaussian_process.kernels import Hyperparameter
from sklearn.gaussian_process import GaussianProcessRegressor
import warnings
warnings.filterwarnings('ignore')
```

```
import missingno as msn
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import plotly.graph_objects as go
from ipywidgets import interact, widgets

np.random.seed(1)

df = pd.read_csv("/content/weatherAUS.csv")
df.dropna(inplace=True)

df_array = np.asarray(df)

date = df_array[0:177,0]
MinTemp = df_array[0:177,2]
MaxTemp = df_array[0:177,3]
Rainfall = df_array[0:177,4]
windspeed = df_array[0:177,8]
humidity = df_array[0:177,13]

y = np.asarray([MinTemp,MaxTemp,windspeed,Rainfall,humidity]).T

X = np.atleast_2d([1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42 ,43 ,44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54 ,55, 56, 57 ,58 ,59, 60, 61, 62, 63 ,64, 65, 66 ,67,
68, 69, 70 ,71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85 ,86, 87 ,88, 89, 90,
91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109,
110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120 ,121, 122, 123, 124, 125, 126,
127, 128 ,129, 130, 131, 132, 133, 134, 135, 136, 137 ,138 ,139, 140, 141, 142, 143,
```

```
144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160,
161, 162, 163, 164, 165 ,166 ,167, 168, 169 ,170, 171, 172, 173 ,174, 175, 176,
177]).T

for i in range(1, 178):
  print(i, end = " ")

x= np.atleast_2d(np.linspace(1,177,177)).T #10739



#model_1

kernel = C()*Exp(length_scale=24,periodicity= 1)
gp = GaussianProcessRegressor(kernel = kernel ,normalize_y=True)

#fitting model_!
gp.fit(X,y)

print("\nLearned Kernal:", gp.kernel_)
print("Log-Marginal-likelihood: %.3f" % gp.log_marginal_likelihood(gp.kernel_.theta))

#prediction
y_pred_1, sigma_1 = gp.predict(x,return_std=True)

#Metrics
from sklearn.metrics import r2_score
#print(y[:,0])
#len(y_pred_1[0:177:,0])
print("r_s:" ,r2_score(y[:,0],y_pred_1[0:177:,0]))
```

```
from sklearn.metrics import mean_squared_error

print("MSE:",mean_squared_error(y[:,0],y_pred_1[0:177:,0]))


from math import sqrt
print("RMSE:", sqrt(mean_squared_error(y[:,0],y_pred_1[0:177:,0])))

#print(y_pred_1[0:177:,3])

#print(y[:,3])
#print(x[:,0])
len(y)

#plot_1
fig = plt.figure(figsize=(15,5))
plt.scatter(X,y[:,0], c = "k",alpha =0.55)
plt.plot(x,y_pred_1[:,0],'b-',label ='predictions')
plt.fill_between(x[:,0], y_pred_1[:,0] - sigma_1[:,0], y_pred_1[:,0] + sigma_1[:,0], alpha
= 0.2, color = "blue", label = "95% confidance interval")#
#plt.xlim(50,60)

#plotting model _1
fig = plt.figure(num = 1, figsize=(11,15), dpi = 300, facecolor="w", edgecolor = "k")

plt.subplot(4,1,1)
plt.plot(X,y[:,0],"r.", markersize=7,label ='Observation')
plt.plot(x,y_pred_1[:,0],'b-',label ='predictions')
```

```
#plt.plot([min(y_pred_1),max(y_pred_1)],    [min(y_pred_1),max(y_pred_1)],    ls="--",
c=".3")
plt.fill_between(x[:,0], y_pred_1[:,0] - sigma_1[:,0], y_pred_1[:,0] + sigma_1[:,0], alpha
= 0.2, color = "blue", label = "95% confidance interval")
plt.xlabel("(a)")
plt.legend(loc="upper right", fontsize = 10)
#plt.ylim(-750,1750)

#model_2

kernel_2 = C()*RQ(length_scale= 24,alpha= 1)
gp = GaussianProcessRegressor(kernel = kernel_2, n_restarts_optimizer=4)

gp.fit(X,y)

y_pred_2 ,sigma_2 = gp.predict(x,return_std=True)

#Metrics
from sklearn.metrics import r2_score
#print(y[:,0])
#len(y_pred_1[0:177:,0])
print("r_s:" ,r2_score(y[:,1],y_pred_2[0:177:,1] ))

from sklearn.metrics import mean_squared_error

print("MSE:",mean_squared_error(y[:,1],y_pred_2[0:177:,1]))


from math import sqrt
print("RMSE:", sqrt(mean_squared_error(y[0:177:,0],y_pred_2[0:177:,1])))
```

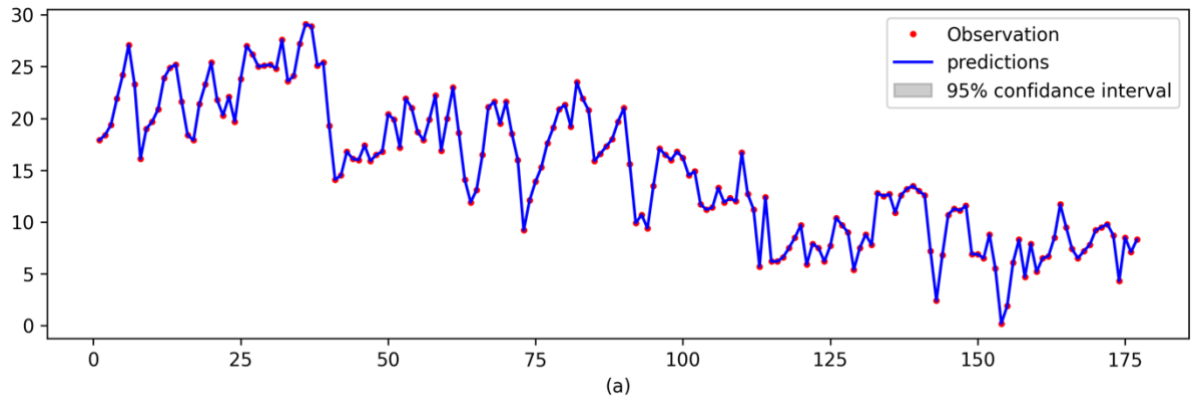
```
fig = plt.figure(num = 1, figsize=(11,15), dpi = 300, facecolor="w", edgecolor = "k")

plt.subplot(4,1,2)
plt.plot(X,y[:,0],"r.", markersize=5,label ='Observation')
plt.plot(x,y_pred_2[:,0],'b-',label ='predictions')
#plt.plot([min(y_pred_1),max(y_pred_1)],  [min(y_pred_1),max(y_pred_1)],  ls="--",
c=".3")
plt.fill_between(x[:,0],  y_pred_2[:,0]  -  1.96*sigma_2,  y_pred_2[:,0]  +  1.96*sigma_2,
alpha = 0.2, color = "k", label = "95% confidance interval")
plt.xlabel("(a)")
plt.legend(loc="upper right", fontsize = 10)

#model_3
from sklearn.gaussian_process import GaussianProcessRegressor
kernel_3  =  C()*RBF(length_scale=4,  length_scale_bounds=(1e-
5,2))*(RQ(length_scale=1,alpha  =  0.5,length_scale_bounds=(1e-
05,2),alpha_bounds=(1e-05,100000.0))+ Exp(length_scale=24, periodicity = 1))
gp = GaussianProcessRegressor(kernel = kernel_3, n_restarts_optimizer=4)
gp.fit(X,y)

y_pred_3 ,sigma_3 = gp.predict(x,return_std=True)

fig = plt.figure(num = 1, figsize=(11,15), dpi = 300, facecolor="w", edgecolor = "k")

plt.subplot(4,1,2)
plt.plot(X,y[:,0],"r.", markersize=5,label ='Observation')
plt.plot(x,y_pred_3[:,0],'b-',label ='predictions')
#plt.plot([min(y_pred_1),max(y_pred_1)],  [min(y_pred_1),max(y_pred_1)],  ls="--",
c=".3")
```

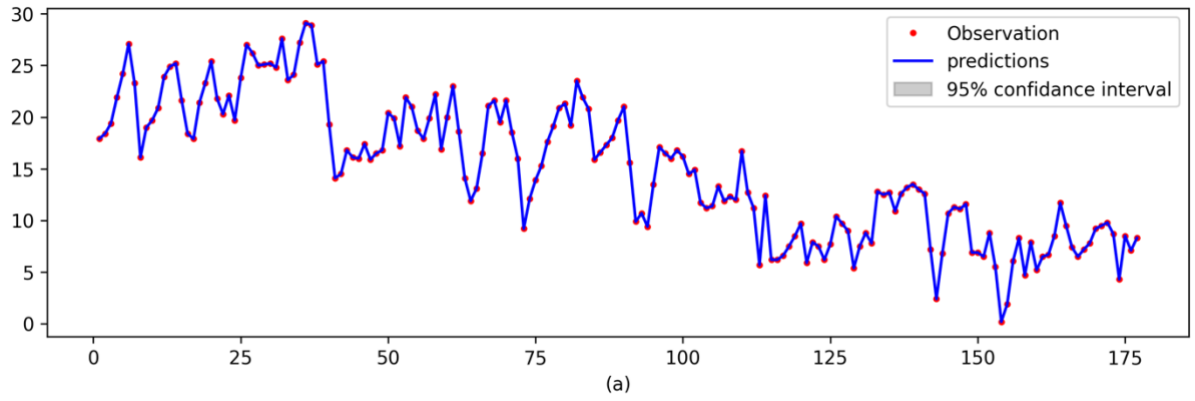
```
plt.fill_between(x[:,0], y_pred_3[:,0] - 1.96*sigma_3, y_pred_3[:,0] + 1.96*sigma_3,
alpha = 0.2, color = "k", label = "95% confidance interval")
plt.xlabel("(a)")
plt.legend(loc="upper right", fontsize = 10)

# model_4
from sklearn.gaussian_process import GaussianProcessRegressor
kernel_4 = C()*Exp(length_scale=24,periodicity=1)*RQ(length_scale=24,alpha=0.5,
length_scale_bounds=(1e-05,2),alpha_bounds=(1e-05,100000.0))
gp = GaussianProcessRegressor(kernel = kernel_4, n_restarts_optimizer=4)
gp.fit(X,y)

y_pred_4 ,sigma_4 = gp.predict(x,return_std=True)

fig = plt.figure(num = 1, figsize=(11,15), dpi = 300, facecolor="w", edgecolor = "k")

plt.subplot(4,1,2)
plt.plot(X,y[:,0],"r.", markersize=7,label ='Observation')
plt.plot(x,y_pred_4[:,0],'b-',label ='predictions')
plt.fill_between(x[:,0], y_pred_4[:,0] - 1.95*sigma_4, y_pred_4[:,0] + 1.95*sigma_4,
alpha = 0.2, color = "k", label = "95% confidance interval")
plt.xlabel("(a)")
plt.legend(loc="upper right", fontsize = 10)

#model_5

from sklearn.gaussian_process.kernels import DotProduct, WhiteKernel
kernel_5 = DotProduct() + WhiteKernel()

gp = GaussianProcessRegressor(kernel = kernel_5, n_restarts_optimizer=4)
```

```
gp.fit(X,y)

y_pred_5 ,sigma_5 = gp.predict(x,return_std=True)

fig = plt.figure(num = 1, figsize=(11,15), dpi = 300, facecolor="w", edgecolor = "k")

plt.subplot(4,1,2)
plt.plot(X,y[:,0],"r.", markersize=7,label ='Observation')
plt.plot(x,y_pred_5[:,0],'b-',label ='predictions')
plt.fill_between(x[:,0],  y_pred_5[:,0]  -  1.95*sigma_5,  y_pred_5[:,0]  +  1.95*sigma_5,
alpha = 0.2, color = "k", label = "95% confidance interval")
plt.xlabel("(a)")
plt.legend(loc="upper right", fontsize = 10)

"""method 2 with numpy"""

n = 10
Xtest = np.linspace(1,10,n).reshape(-1,1)

# Define the kernel function
```

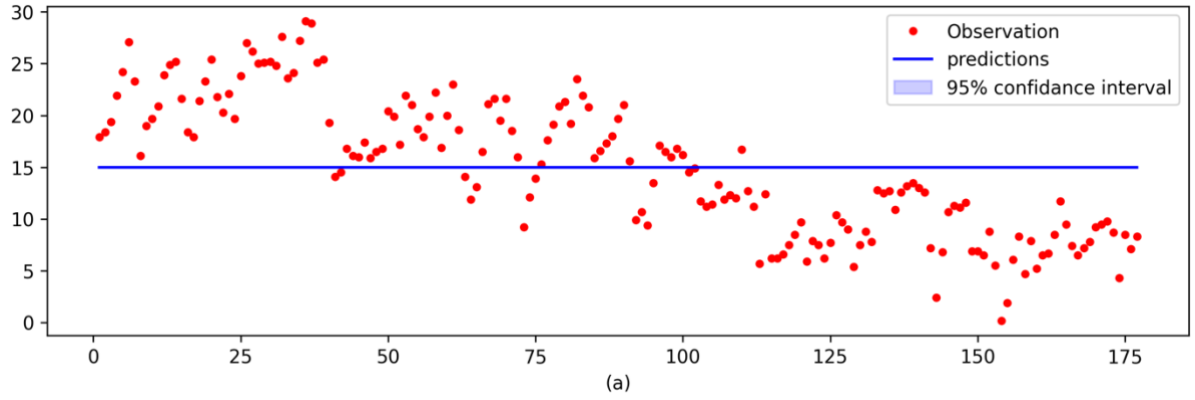
```
def kernel(a, b, param):
    sqdist = np.sum(a**2,1).reshape(-1,1) + np.sum(b**2,1) - 2*np.dot(a, b.T)
    return np.exp(-.5 * (1/param) * sqdist)

param = 0.1
K_ss = kernel(Xtest, Xtest, param)

# Get cholesky decomposition (square root) of the
# covariance matrix
L = np.linalg.cholesky(K_ss + 1e-15*np.eye(n))
# Sample 3 sets of standard normals for our test points,
# multiply them by the square root of the covariance matrix
f_prior = np.dot(L, np.random.normal(size=(n,3)))

# Now let's plot the 3 sampled functions.
plt.plot(Xtest, f_prior)
#plt.axis([0,11, -3, 3])
plt.title('Three samples from the GP prior')
plt.show()

# Noiseless training data
Xtrain = np.array([5, 4, 3, 2, 1]).reshape(5,1)
ytrain = np.sin(Xtrain)

# Apply the kernel function to our training points
K = kernel(Xtrain, Xtrain, param)
L = np.linalg.cholesky(K + 0.00005*np.eye(len(Xtrain)))

# Compute the mean at our test points.
K_s = kernel(Xtrain, Xtest, param)
```

```
Lk = np.linalg.solve(L, K_s)
mu = np.dot(Lk.T, np.linalg.solve(L, ytrain)).reshape((n,))

# Compute the standard deviation so we can plot it
s2 = np.diag(K_ss) - np.sum(Lk**2, axis=0)
stdv = np.sqrt(s2)

# Draw samples from the posterior at our test points.
L = np.linalg.cholesky(K_ss + 1e-6*np.eye(n) - np.dot(Lk.T, Lk))
f_post = mu.reshape(-1,1) + np.dot(L, np.random.normal(size=(n,3)))

plt.plot(Xtrain, ytrain, 'bs', ms=10)
plt.plot(Xtest, f_post)
plt.gca().fill_between(Xtest.flat, mu-2*stdv, mu+2*stdv, color="#dddddd")
plt.plot(Xtest, mu, 'r--', lw=2)
#plt.axis([-5, 3, -3, 3])
plt.title('Three samples from the GP posterior')
plt.show()
```

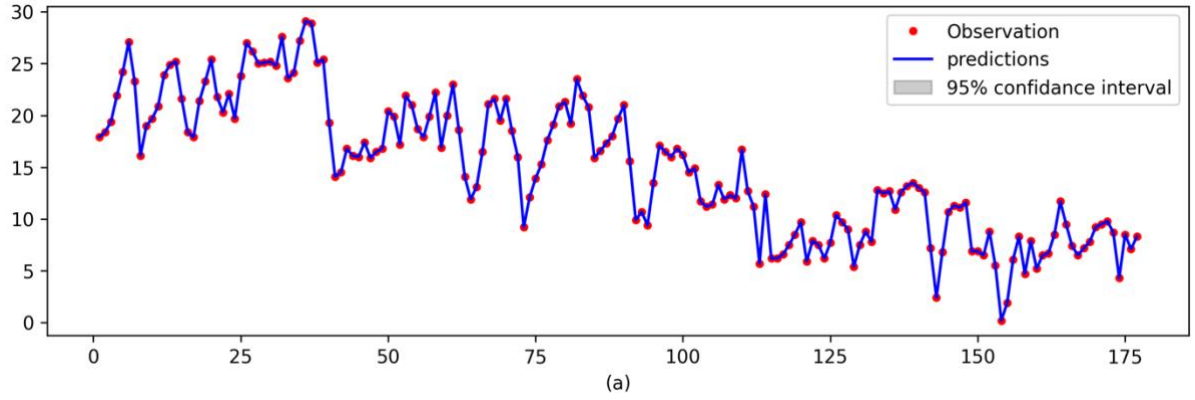



Observation
predictions
95% confidance interval

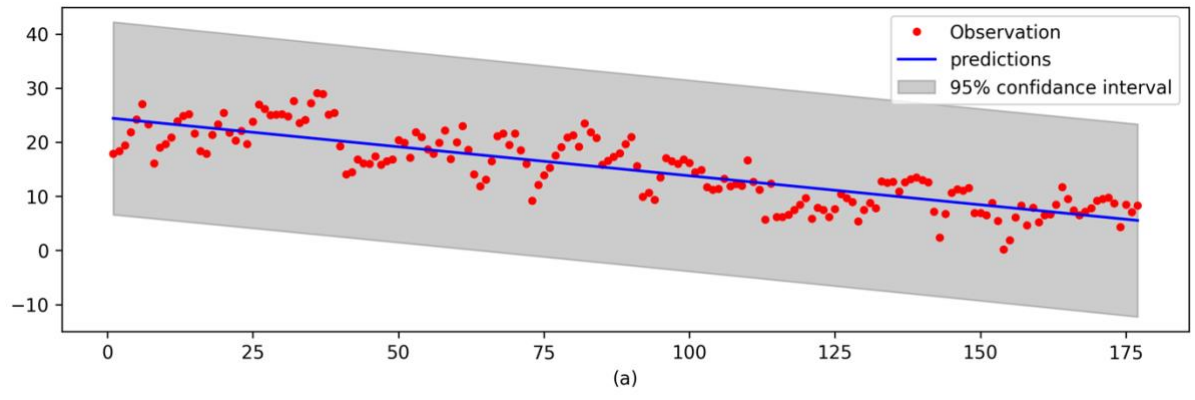
(a)


Observation
predictions
95% confidance interval

(a)

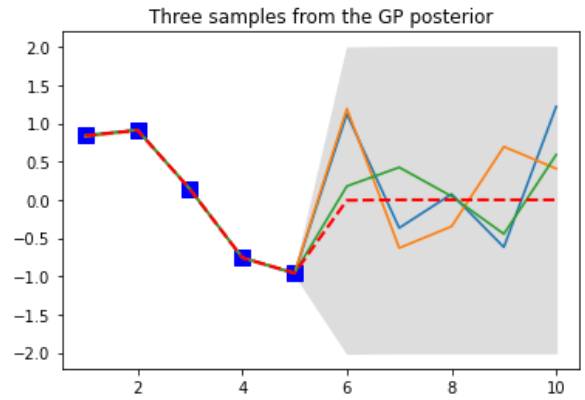

Observation
predictions
95% confidance interval

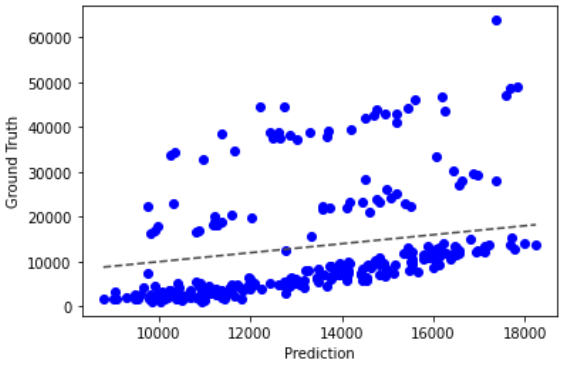
(a)

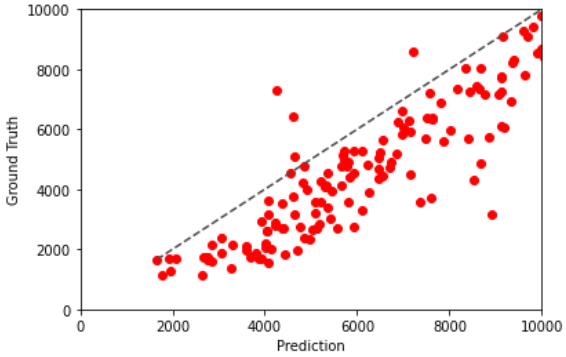

Observation
predictions
95% confidance interval

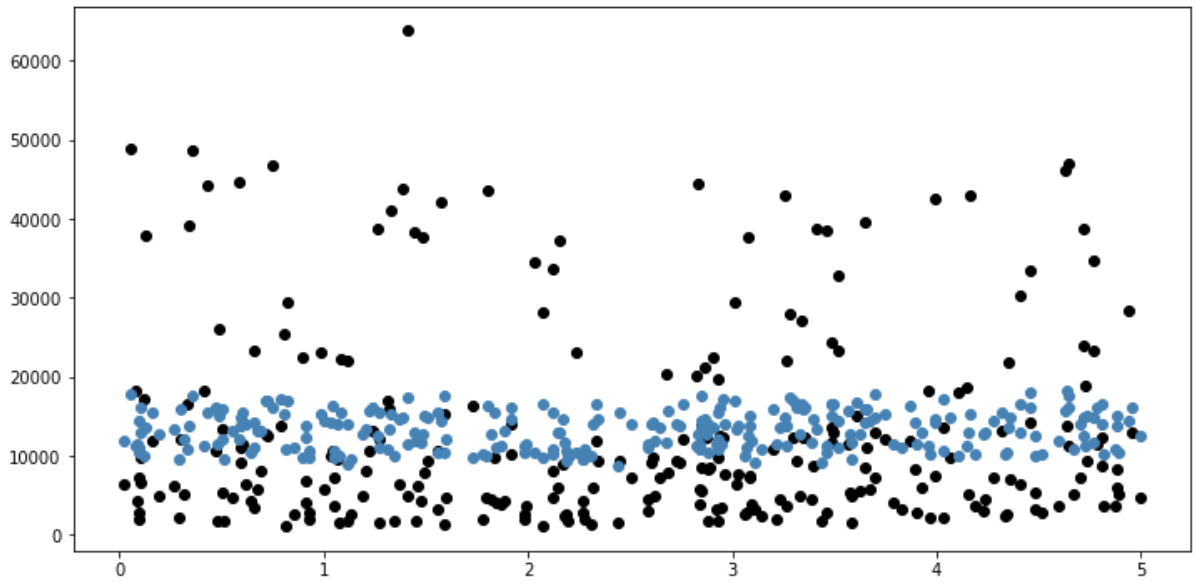
(a)

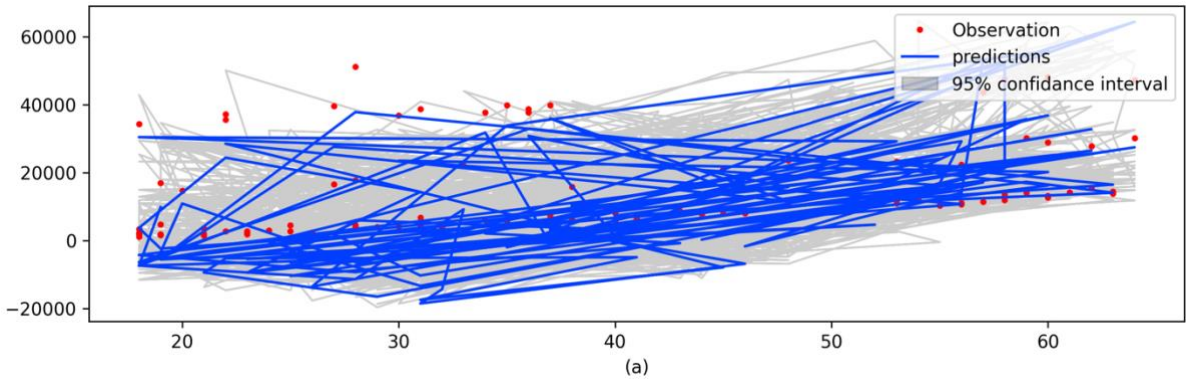

Three samples from the GP prior


Three samples from the GP posterior


Ground Truth
Prediction


Ground Truth
Prediction




Observation
predictions
95% confidance interval
60000
40000
20000
0
−20000
20
30
40
50
60
(a)

Fuzzy Logic Controller and Evolutionary Algorithms for Assistive Care Home

Dhanasekaran, Roshan

12135637
Faculty of Engineering, Environment, and Computing
MSc Data Science and Computational Intelligence, Coventry University
dhanasekar@uni.coventry.ac.uk

Abstract – In this paper fuzzy logic controller was implemented in building an assistive care home for disabled people. Appropriate inputs to control the corresponding outputs were developed using logical and mathematical rules. Mamdani fuzzy interface was used from MATLAB fuzzy toolbox. upon building the Fuzzy logic controller optimization was performed by using a genetic algorithm in the global optim toolbox and the outputs of the algorithms were compared. It was found that the evolutionary algorithms have an high influence on improving the performance of our fuzzy logic controller (FLC)

Keywords: Mamdani Fuzzy interface, optimization, MATLAB, Assistive care home, generic algorithm

I. INTRODUCTION

Having an automated home is a necessity considering the wild growth of technology in the past 10 years and especially in the field of machine learning and AI. We should be able to develop a house which is nature friendly and also cost-friendly and reduces the carbon footprint which addresses a much bigger problem. In this paper we are trying to solve a subproblem which is building an assistive home for disabled people, this can be achieved by collecting data from sensors which monitor the temperature blood pressure, super level and gestures to name a few with this input data we can build a fuzzy logic system which can automate the outputs based on the inputs. On successfully building an effective fuzzy logic controller for a house this information can be used to build an FLC for an entire flat. There are mainly two fuzzy interfaces namely Mamdani fuzzy system and Sugeno fuzzy system. In this paper, we intend to use Mamdani fuzzy interface to perform all the tasks. In building an optimum FLC we intend to use evolutionary algorithms to optimize our FLC for which Global Optim Toolbox was made use. The genetic algorithms help to remove the limitations of non-linearity which will inter give us a better FLC for our Assistive care home.

III. ASSISTIVE CARE HOME FUZZY DESIGN

It can be very tedious to build a common system for people who can find a certain temperature cold and a certain temperature not so cold, manual operation is not feasible in a disabled home. To build a sustainable fuzzy system we need to consider temperature, humidity, CO2, blood pressure, sugar levels and hand gestures as our inputs. Considering all the above-discussed factors building an efficient model can be a game changer. The fuzzy logic can exactly solve this issue by taking the input values and taking the decision based on the rules given to each membership function. The below figure represents the outline of a fuzzy system.

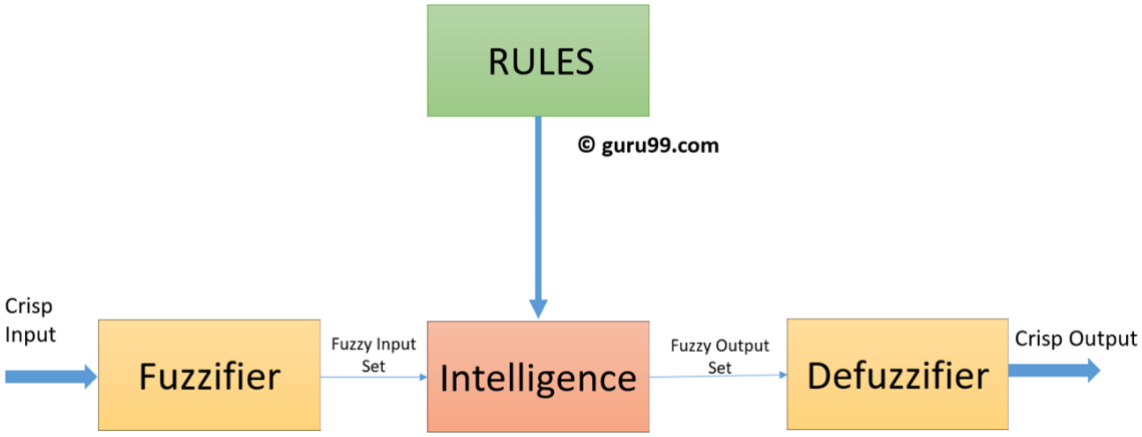


Figure 1: Assistive care home fuzzy design

The above figure explains the basic function of an FLC where the fuzzifier gets all the crisp inputs and an interface either can be Mamdani/Sugano with the set of rules and give us the outputs. Those outputs need to be defuzzified to get crisp outputs. Each component is explained in detail below.

A) Inputs and membership functions

There are seven inputs in our FLC which play a major role in constructing an effective assistive home fuzzy logic system namely; temperature, humidity, CO2, lights, blood pressure, sugar levels, and hand gestures. By assigning the rules for these parameters an FLC can be developed, for each parameter, a membership function was defined and the membership function or either triangle or trapezoidal. The overview of each input is as follows.

1) Temperature

Temperature plays a huge role in controlling the room temperature from the water temperature. Therefore assigning the right temperature scale is vital. Temperature can be divided into very cold (VC), mild cold (MC), Average temperature (AT), Hot (H), and very hot (VH). We have used a triangular function for VC and VH, a trapezoidal function for MC, AT and H

Table 2 Temperature membership function

Temperature Membership Function

S.NO	Membership function	Range	Type
1	VC	-50 to -8	Triangle
2	MC	-8 to 6	Trapezoid
3	AT	2 to 18	Trapezoid
4	H	16 to 29	Trapezoid
5	VH	25 to 60	Triangle

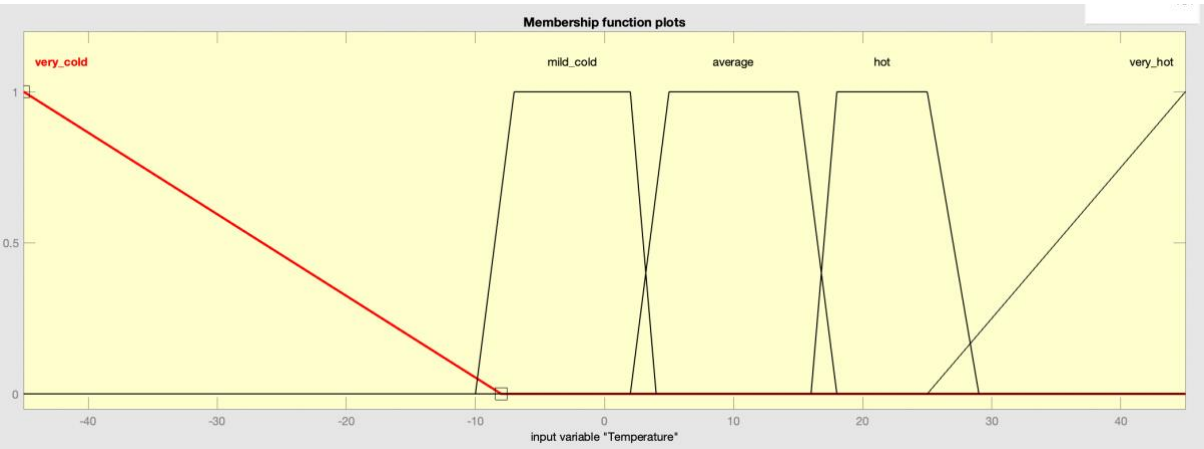


Figure 1: Temperature membership function

2) Humidity

Another major factor needed to be considered is the humidity it is the amount of water vapour in the air inside the room and outside considering this we can decide whether to keep the windows open or if there is less humidity we don't want to person to have dry skin or can have other issues. We consider the relative humidity which builds our membership function from the range of 0 to 100. We have their membership functions namely low humidity (LH), Average Humidity (AH), and high humidity (HH).

Table 3 Humidity membership function

Humidity Membership Function			
S.NO	Membership function	Range	Type
1	LH	-37 to 25	Trapezoid
2	AH	19 to 60	Trapezoid
3	HH	55 to 100	Trapezoid

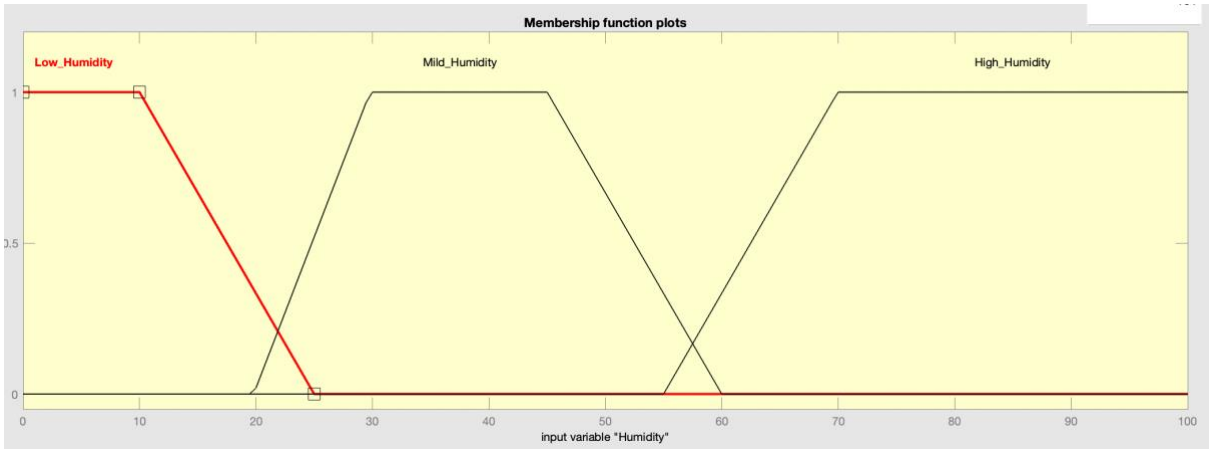


Figure 2: Humidity membership function

The above table depicts the inputs given to control the humidity, the humidity can be maintained with the help of coolers ad heaters.

3) CO2 Sensor

The CO2 sensors can help us to determine the cO2 levels inside the house, if there is an increase in CO2 levels the windows and doors needs to be opened to maintain the O2 levels this will intern help the person to breathe fresh air and keep him healthy. The CO2 levels are separated across low CO2 (L CO2), Mild CO2 (M CO2), Average CO2 (A CO2) and high CO2 (H CO2). The table and the figure are shown below. The unit of measure is ppm.

Table 4 CO2 Membership function

CO2 Membership Function			
S.NO	Membership function	Range/ppm	Type
1	L CO2	0 to 200	Trapezoid
2	M CO2	148 to 348	Trapezoid
3	A CO2	300 to 600	Trapezoid
4	H CO2	550 to 1375	Trapezoid

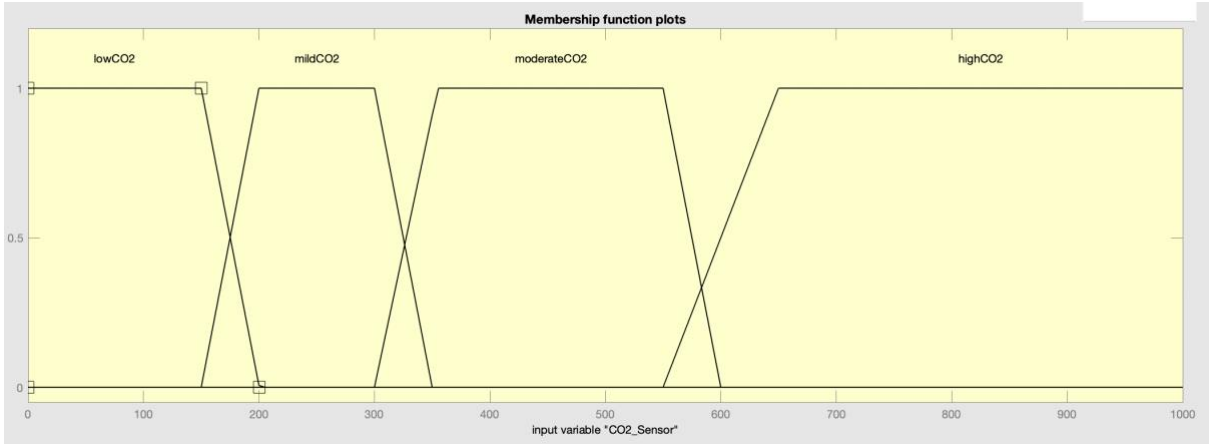


Figure 3: CO2 Membership function

4) Heart Rate

Since we are building and smart system for disabled people it is necessary to have constant supervision over the heart rate if any change there is any increase or decrease in the regular heart rate the emergency phone needs to contact the nearest hospital. The heart rate can be separated into 3 membership functions namely low heart rate (L HR), normal heart rate (N HR) and high heart rate (H HR)

Table 5 Heart rate membership function

Heart Rate Membership Function			
S.NO	Membership function	Range/bpm	Type
1	L HR	0 to 60	Trapezoid
2	M HR	50 to 105	Trapezoid
3	H HR	100 to 160	Trapezoid

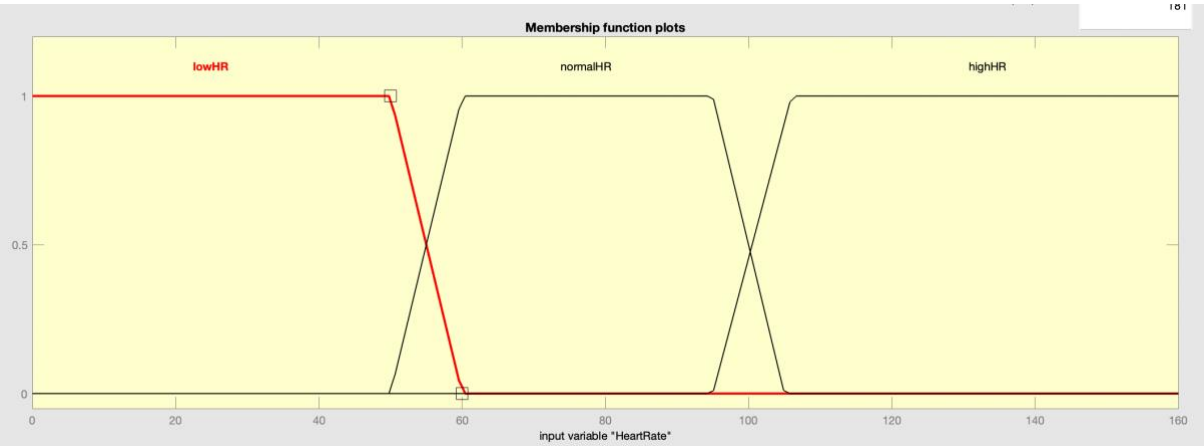


Figure 4: Heart rate membership function

5) Blood pressure

Sugar levels and blood pressure needs to be constantly monitored like the heart rate if in case there is low BP the person can be treated immediately by calling the nearest hospital in seconds and the problem can be addressed. The blood pressure can be separated into three membership functions low BP (L BP) normal BP (N BP) and high BP (H BP). The below figure and the table represents the membership function of blood pressure.

Table 6 Blood pressure membership function

Blood pressure Membership Function			
S.NO	Membership function	Range/mmHg	Type
1	L HR	0 to 80	Trapezoid
2	M HR	80 to 100	Trapezoid
3	H HR	95 to 257	Trapezoid

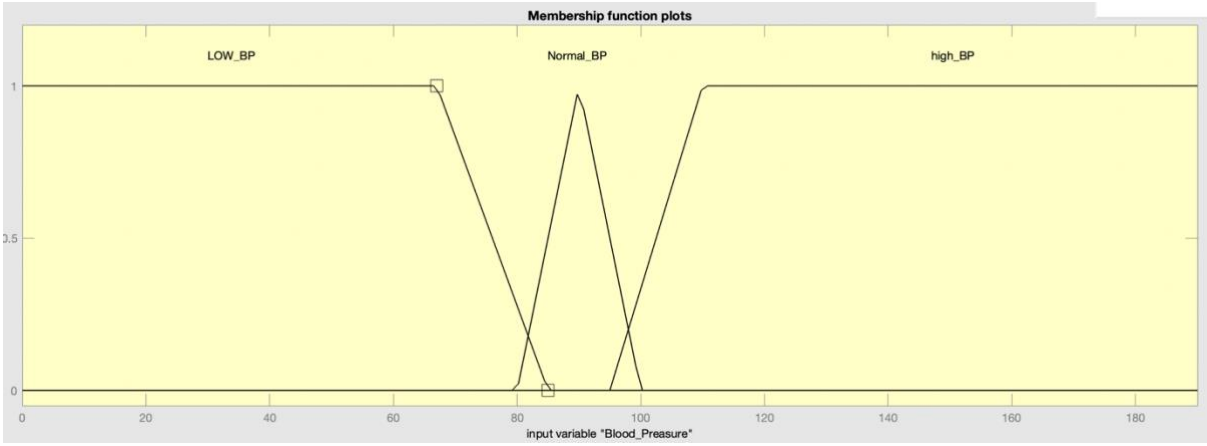


Figure 5 Blood pressure membership function

6) Light

Light from the sun or any other source can be used to determine whether the window or the blinds needed to be opened or shut. The amount of light can also indicate the temperature outside and based on the time and the amount of light we can decide whether to turn on the LED or not. The light parameter can be separated into their membership functions namely dim light (DL), moderate light (ML) and bright light (BL).

Table 7 Light Membership function

light Membership Function			
S.NO	Membership function	Range/Lux	Type
1	L HR	0 to 1500	Trapezoid
2	M HR	1000 to 2000	Trapezoid
3	H HR	2673 to 4000	Trapezoid

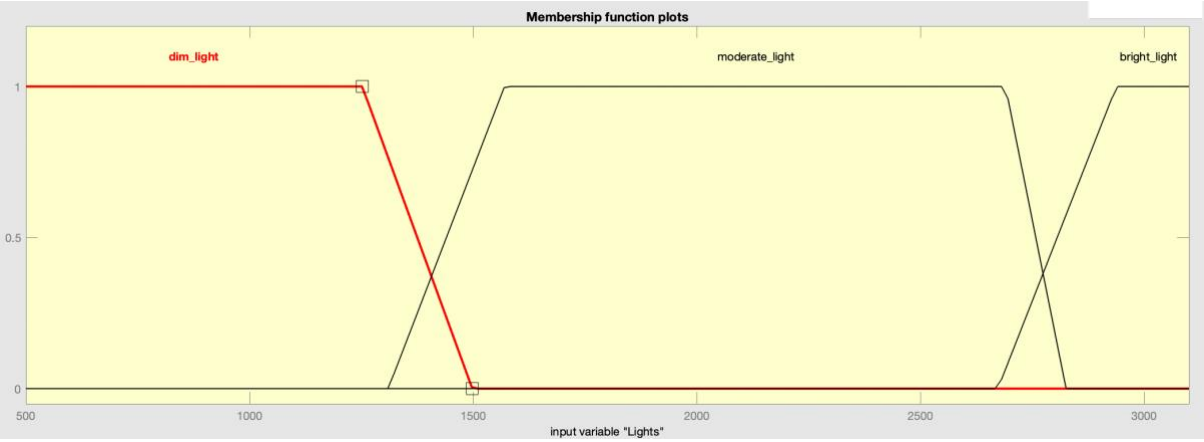


Figure 6: Blood pressure membership function

7) Hand gesture

This input function was introduced to reduce the stress of the person moving around the house instead he/she can control the TV volume or music Volume just from the moment of their hands. The membership function and namely hand low distance from the head (HLD), hand

average distance from the head (HMD), hand far from the head (HLD). The unit of measure is cm. with is information the below membership table can be generated.

Table 8 hand gestures membership function

Hand gestures Membership Function			
S.NO	Membership function	Range/cm	Type
1	HLD	0 to 30	Triangle
2	HMD	30 to 70	Triangle
3	HLD	70 to 100	Triangle

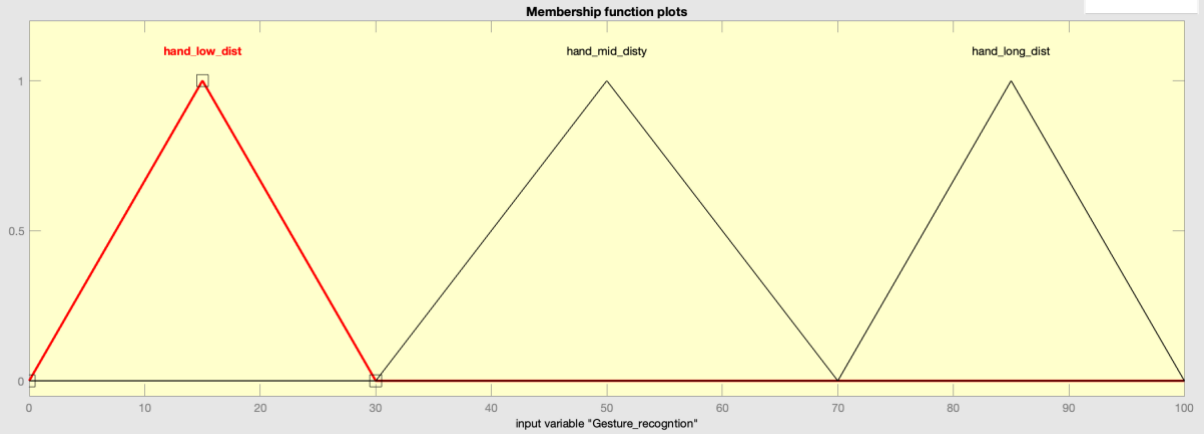


Figure 7 hand gestures membership function

B) Fuzzy Logic Interface

The brain of the fuzzy system is a fuzzy interface. The interface uses the set theory to map the inputs to outputs. The two main fuzzy interfaces are Mamdani and Sugeno fuzzy, in the interface this paper will use the Mamdani interface to map the inputs to outputs. The Mamdani system was developed in 1975 with some simple rules of fuzzification, rule inference, aggregation and defuzzification.

A crisp set of inputs are passed through and converted into fuzzy sets with fuzzification. The next step is to use fuzzy operators (AND/OR) is used to obtain a single output. These numbers are then applied to their corresponding membership function. Mamdani follows a particular way to defuzzify them by following the above methods we can have a crisp output with which we can take a certain decision. MATLAB fuzzy toolbox was used to develop the fuzzy system. The below figure shows the body of the assistive care home system.

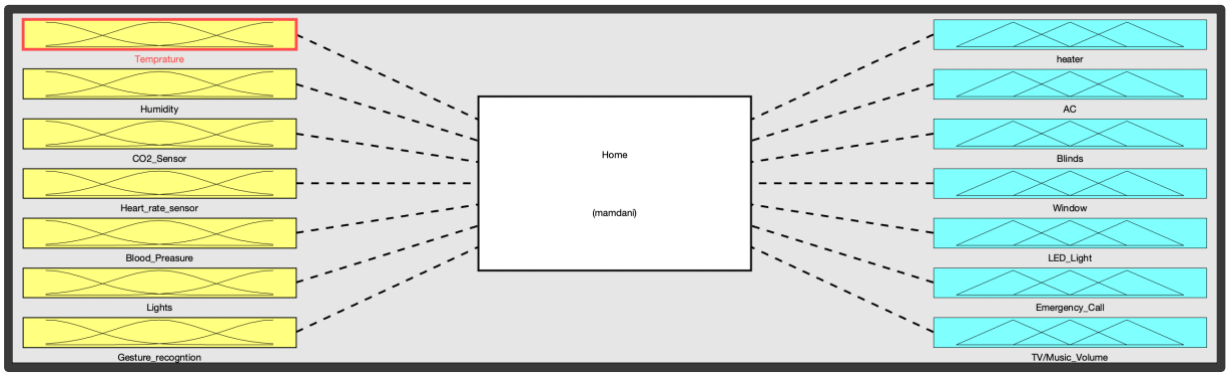


Figure 8 Mamdani Fuzzy Logic Interface

C) Output Membership function

As shown in the above figure like the input membership function we have a corresponding output membership function. We also assign a range for each membership function.

1) Heater

The range for the heaters is taken from 0 to 5 this can be used to maintain the temperature and humidity in the home and the water supply. The three membership functions are heater temperature low (HTL), heater temperature high (HTH) heater temperature medium (HTM). The membership function is shown in the below table.

Table 9 heater membership function

Heater Membership Function			
S.NO	Membership function	Range	Type
1	HTL	0 to 1.5	Trapezoid
2	HTM	1.5 to 3.5	Trapezoid
3	HTH	3.5 to 5	Trapezoid

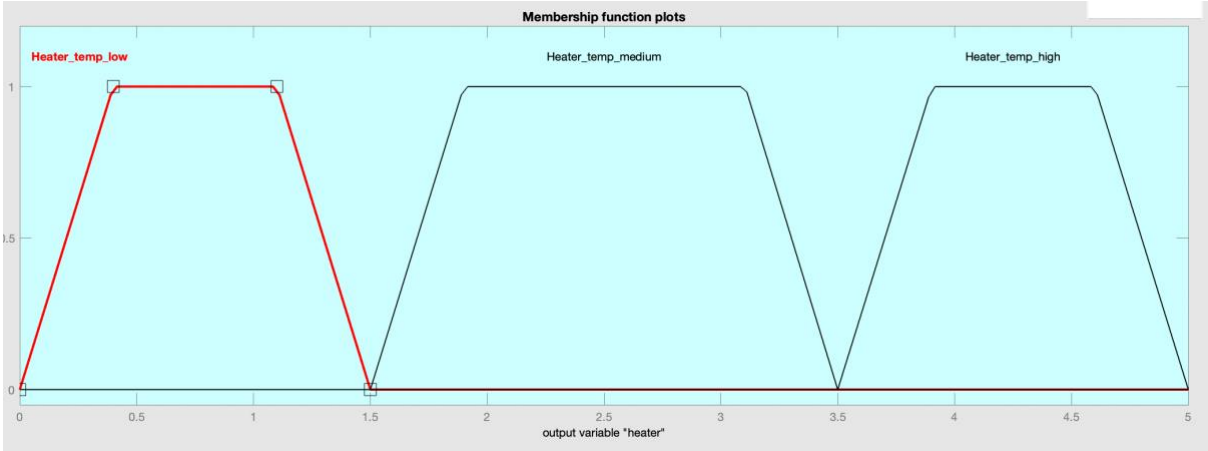


Figure 9 heater membership function

2) AC

The air conditioner can be tuned based on the input temperature and humidity. The range of the AC is from 0 – 5 and has three membership functions AC temperature low (ATL), AC temperature medium (ATM) and AC temperature high (ATH).

Table 10 Ac membership function

AC Membership Function			
S.NO	Membership function	Range	Type
1	ATL	0 to 1.5	Trapezoid
2	ATM	1.5 to 3.5	Trapezoid
3	ATH	3.5 to 5	Trapezoid

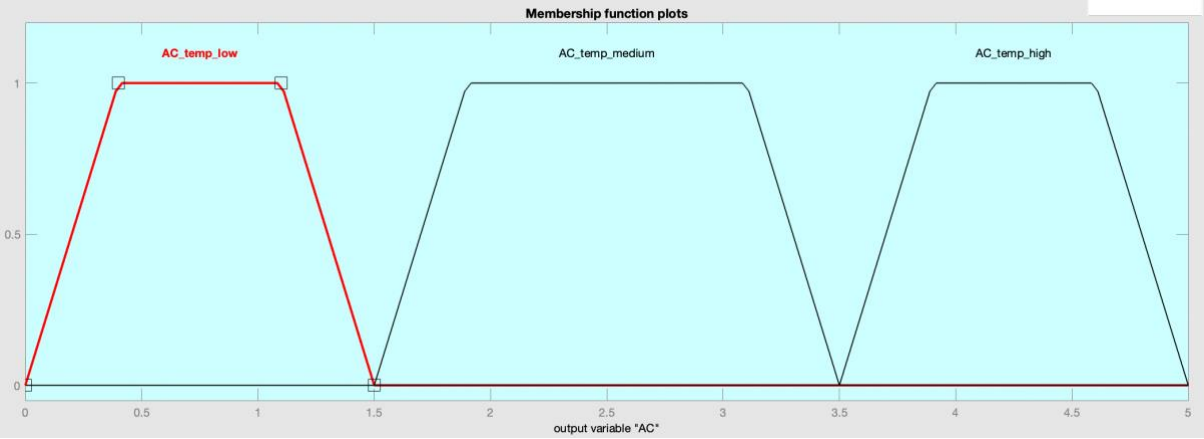


Figure 10 Ac membership function

3) Blinds

The blinds can be controlled by the intensity of light getting into the house therefore the blinds can have three membership functions closed blinds (CB), partial Blinds (PB), and open blinds (OB). The below figure and table better describes the membership function

Table 11 Blinds membership function

AC Membership Function			
S.NO	Membership function	Range	Type
1	CB	0 to 0.35	Trapezoid
2	PB	0.25 to 0.75	Trapezoid
3	OB	0.69 to 1	Trapezoid

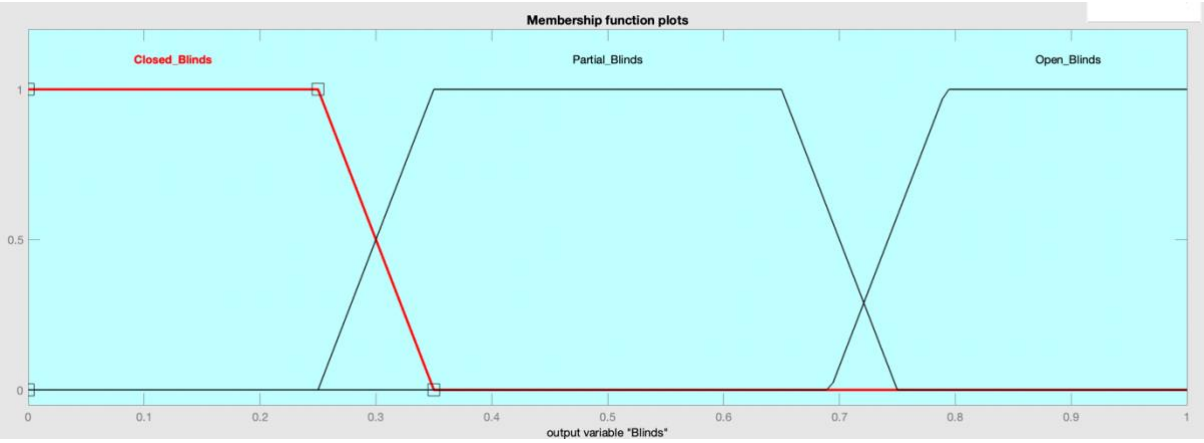


Figure 11 Blinds membership function

4) Window

Like the blinds, the windows can be controlled based on the temperature outside and the amount of light getting inside the house and also if the sun is setting or rising. The window has three membership functions, closed window (CW), partial window (PW) and open window (OW).

Table 12 Window membership function

Window Membership Function			
S.NO	Membership function	Range	Type
1	ATL	0 to 3.5	Trapezoid
2	ATM	0.28 to 0.78	Trapezoid
3	ATH	0.71 to 1.07	Trapezoid

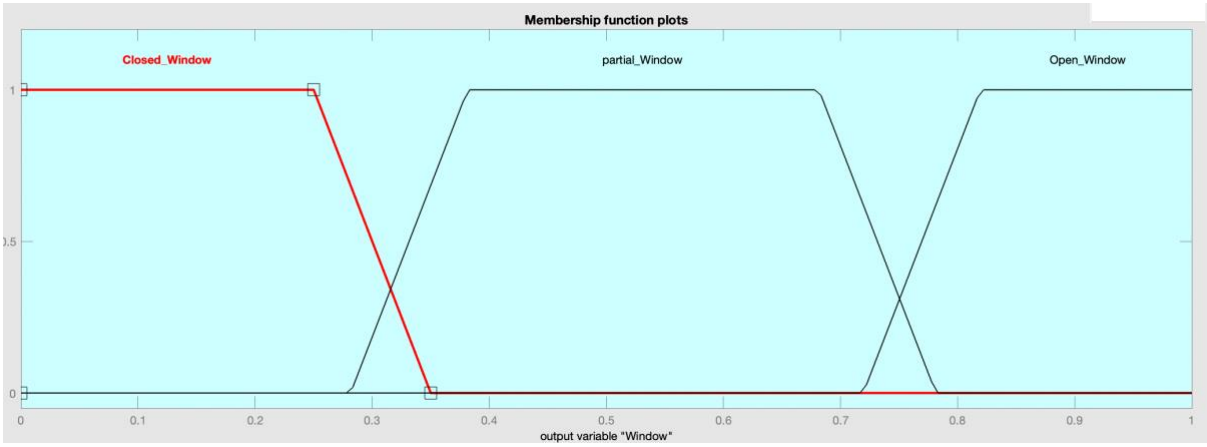


Figure 12 Window membership function

5) LED lights

The LED light also has three membership functions based on the light intensity and time of the day the LED light can be adjusted. The three membership functions are Low (L), Average (A), bright (B)

Table 13 LED Membership function

LED Membership Function			
S.NO	Membership function	Range	Type
1	L	0 to 0.4	Triangle
2	A	0.2 to 0.8	Triangle
3	B	0.6 to 1	Triangle

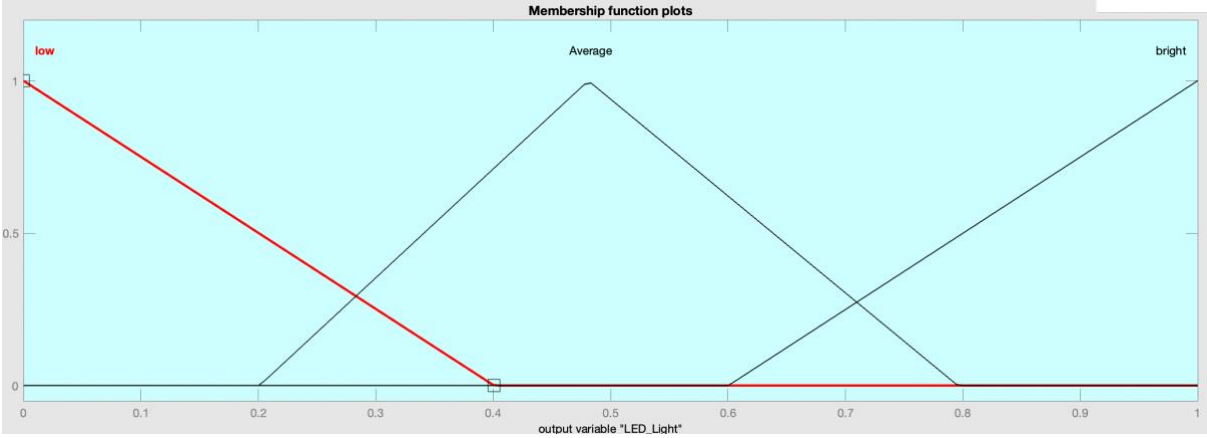


Figure 13: LED Membership function

6) Emergency Call

This output function is extremely useful because it takes inputs from the heart rate sensor and blood pressure sensor and if there is any anomaly in the reading an emergency call can be made. The emergence call is separated into three membership functions namely don't call (DC) call nurse (CN) and call an ambulance (CA)

Table 14 Emergency call membership function

Emergency Membership Function			
S.NO	Membership function	Range	Type
1	DC	0 to 0.3	Triangle
2	CN	0.3 to 0.75	Triangle
3	CA	0.6 to 1	Trapezoid

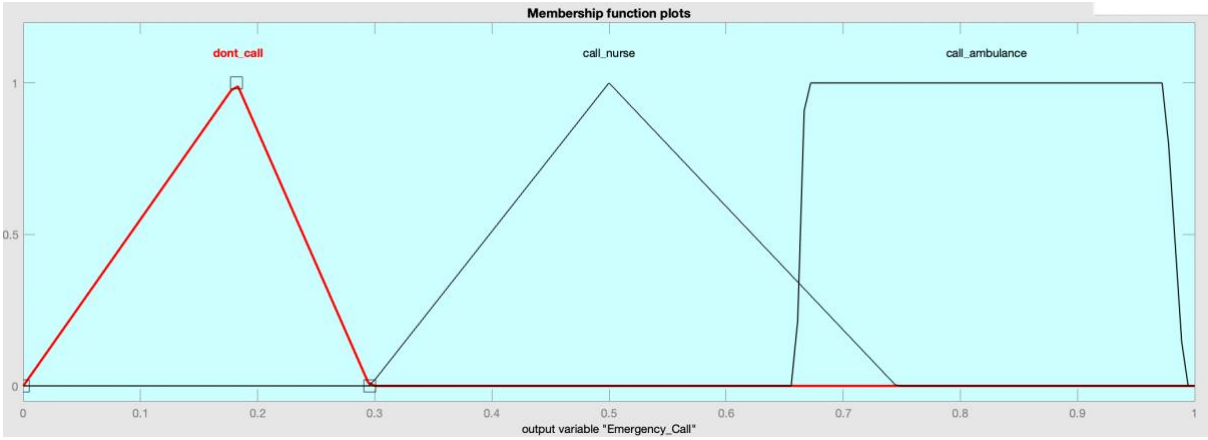


Figure 14:Emergency call membership function

7) TV/Music Volume

The final output parameter is the volume for TV and music with the help of gestures from our hands say when the hands are near to the head turn the volume bit low and if the hand is farther away from the head then reduce the volume by 50% and the hand is away from the head then reduce the volume fully. The membership functions are L, M, and H.

Table 15 TV/ Music volume

Emergency Membership Function			
S.NO	Membership function	Range	Type
1	L	0 to 8	Triangle
2	M	2 to 18	Triangle
3	H	12 to 20	Triangle

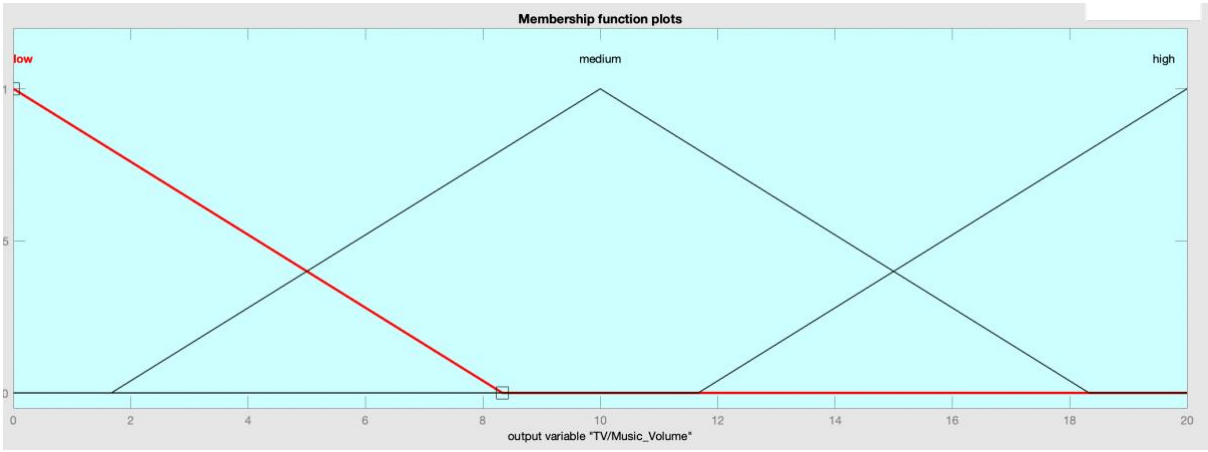


Figure 15 TV/ Music volume membership function

D) Fuzzy Rules

The fuzzy rules are built based on the relationship between the input and the output. A domain expert can help us find the relation between the input and the outputs but in our case of building

an assistive care home, it is pretty straightforward. With the precise implementation of the rules, we can expect a good fuzzy logic controller as the output. The rules can connect one input with different output parameters or the vies versa, it follows the t-norm or the t-conorms, where t – norms use AND gate and t- conforms uses the OR operator. the rules for various input and output functions are shown below.

E) Aggregation

As the name describes aggregation is the process of unifying all the given set of rules. All the rules are reviewed and given a single output. It takes the union of the given output for the given data. If there are two membership functions with different levels the maximum value will be considered.

F) Defuzzification

In simple terms, the process of Defuzzification is converting the fuzzy output into a crisp output with the help of certain methods. There are several methods for defuzzification, in this paper, we will be using the centre of gravity (COG) method to get the crisp outputs. Upon defuzzification, the rules can be viewed and the below figures represent the rules.

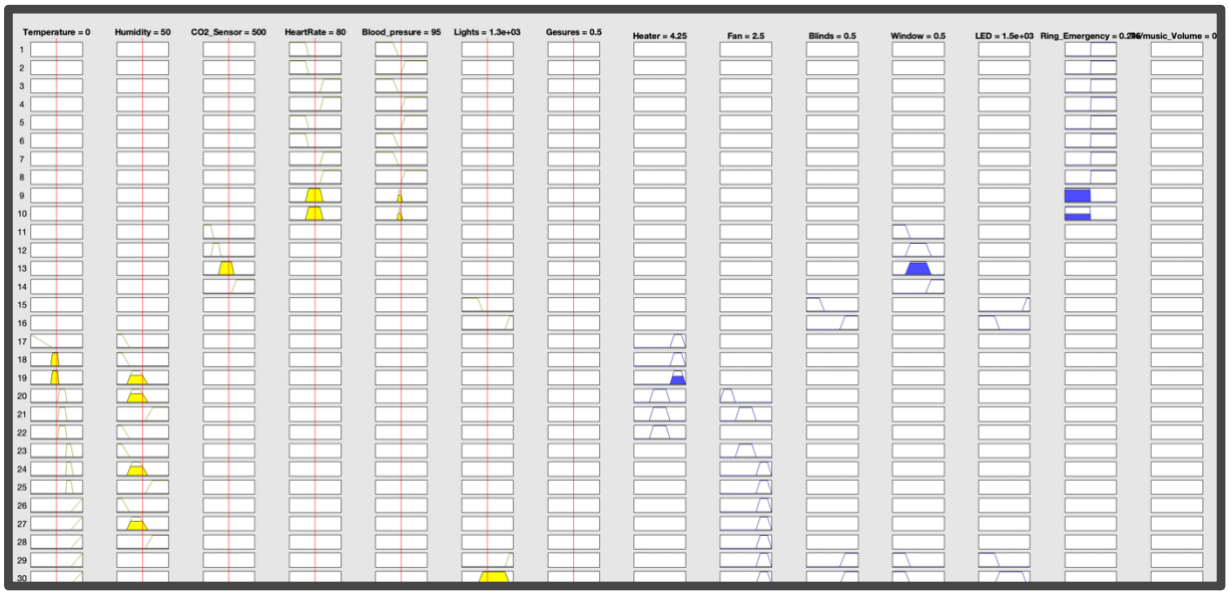


Figure 16:Rules for the Fuzzy model

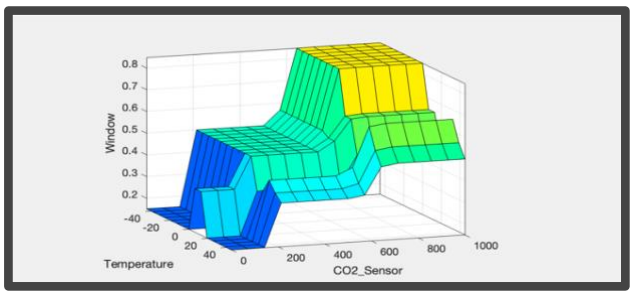


Figure 17 temperature and CO2 vs Window

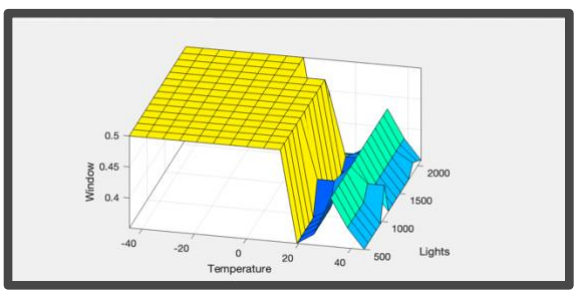


Figure 18 lights and temperature vs window

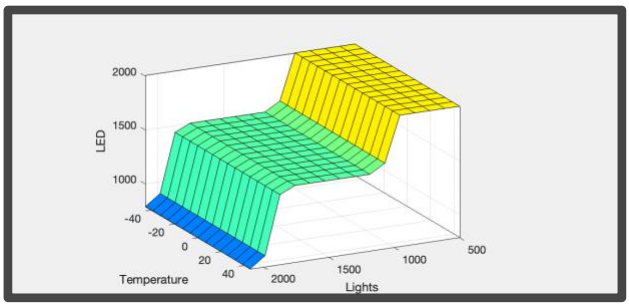


Figure 19 lights and temperature vs LED

IV. GENETIC ALGORITHMS FOR FUZZY SYSTEMS

Genetic algorithms are a branch of algorithms from evolutionary algorithms as a set of algorithms which are derived from natural behaviour like the ant colony and flock of birds. The generic algorithm follows the human gene mutation method and optimizes the problem in our case we are trying to optimize our fuzzy logic controller. We intend to take one input and one output function and find the relevant fitness function and optimize and find the number of generations required to get an optimal solution. A short brief about genetic algorithms is discussed below.

A) Genetic Algorithm

The genetic algorithm was derived from a theory produced by Darwin. The theory clearly states the survival of the fittest and the fittest are allowed to produce offspring's building the next adaptive generation. There are five major steps in this algorithm namely initial population, fitness function selection, Cross over and mutation.

1) Initial population

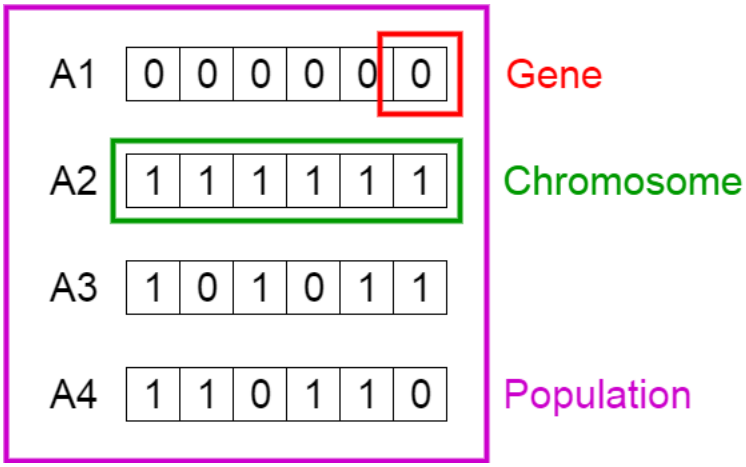


Figure 20 Initial population

As the figure above shows, there are genes represented as 1 and 0 and an entire chromosome which showed in green and the collection and chromosome is called population. In our case the fitness function and the membership function can be optimised accordingly, this will be done by first converting the inputs and the outputs into a binary state.

2) Fitness function

The fitness function can determine the fitness of an individual that can be selected for reproduction. This function will also give us a fitness score where we can determine the level of fitness. All the process was executed in MATLAB using the global optim toolbox for each individual we need to find the fitness function the below are the fitness function for our variable.

```
function y = fitness_func(x)

  y(1) = - 0.1694*x(1) + 7.001; %% temp vs heater
  y(2) = 0.005366*x(1) - 0.2808; %% humidity vs AV
  y(3) = 0.0001301*x(1) + 4.738; %% Ac vs CO2
  y(4) = 111.7*x(1) + 12.84; %% heart rate vs emergancy call
  y(5) = 0.0004902*x(1) + 0.07742 %% lights vs blinds


end
```

Figure 21 few of the fitness function

In the above fitness function, there are 5 values of X which represent temperature, humidity, CO2, light, hand gestures and blood pressure and heart rate, on the other hand, Y represents tall output functions like the Tv volume, windows etc. note: not all the fitness function are shown in the above figure.

3) Selection

According to the fitness score, there will be individuals will be selected to produce offspring. There are many methods where selection can be performed but in this paper, we, do the roulette wheel. The roulette wheel works on the principle that the selection and the fitness are directly proportional to each other. This is one of the most commonly used methods of selection.

4) Cross Over

Cross-over is one of the important steps in the genetic algorithms where the two selected individuals randomly cross over their genes. This helps us to build better and at the same time weaker offspring. This can be specified in the MATLAB toolbox.

5) Mutation

Mutation helps us to carry the strong genes from the previous generation to the new generation and helps to have diversity in the chromosomes. There are many methods to implement mutation in MATLAB, we will be using them to perform mutation. The experiment results are shown in the next section.

B) Experimental Outputs For Genetic Algorithms

The below figure shows the setting we have implemented to get the best results. See figure 21

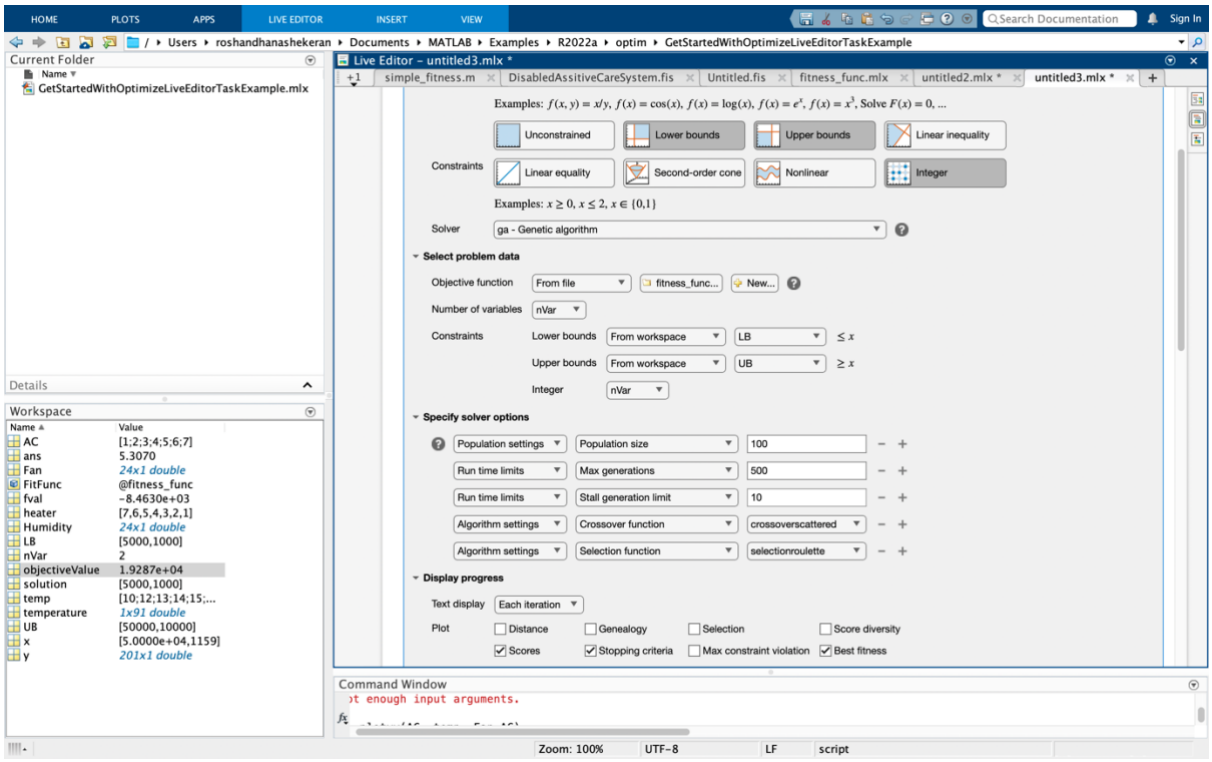


Figure 22 Toolbox setup

1) Temperature vs Heater

```
Single objective optimization:
2 Variable(s)
1 Integer variable(s)

Options:
CreationFcn:       @gacreationuniformint
CrossoverFcn:      @crossoverscattered
SelectionFcn:      @selectionroulette
MutationFcn:       @mutationpower

                                 Best          Mean        Stall
Generation      Func-count     Penalty       Penalty     Generations
     1             200          -8411         -4977          0
     2             298          -8437         -5537          0
     3             396          -8459         -5701          0
     4             494          -8459         -5908          0
     5             592          -8459         -5862          0
     6             690          -8463         -6205          0
     7             788          -8463         -6220          1
     8             886          -8463         -6108          0
     9             984          -8463         -6179          0
    10            1082          -8463         -5791          0
    11            1180          -8463         -5813          0
    12            1278          -8463         -5916          0
    13            1376          -8463         -5832          0
    14            1474          -8463         -6088          0
    15            1572          -8463         -6370          0
    16            1670          -8463         -6450          0
    17            1768          -8463         -5962          0
    18            1866          -8463         -5740          0
    19            1964          -8463         -6222          0
    20            2062          -8463         -5863          0
    21            2160          -8463         -5641          1
    22            2258          -8463         -5856          0
    23            2356          -8463         -5901          0
```

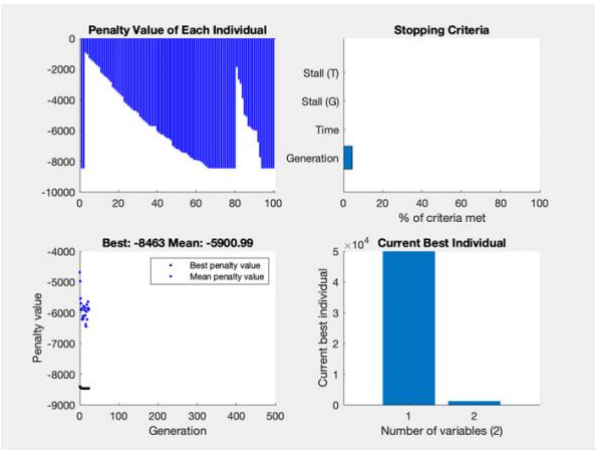


Figure 23 results of the generic algorithms for temperature and heater

Without a proper optimization setting, it took 52 generations to get the optimal results but after the proper optimization, it took only 23 generations to get the required output.

2) Humidity vs AC

```
Single objective optimization:
2 Variable(s)

Options:
CreationFcn:       @gacreationuniform
CrossoverFcn:      @crossoverscattered
SelectionFcn:      @selectionroulette
MutationFcn:       @mutationadaptfeasible

                                 Best          Mean     Stall
Generation      Func-count        f(x)          f(x)    Generations
    1              200           26.55         117.4        0
    2              295           26.55         88.27        1
    3              390           26.55         67.19        2
    4              485           26.55         50.98        3
    5              580           26.55         38.46        4
    6              675           26.55         29.97        5
    7              770           26.55         29.88        6
    8              865           26.55         26.55        7
    9              960           26.55         26.55        8
   10             1055           26.55         26.55        9
   11             1150           26.55         26.55       10
```

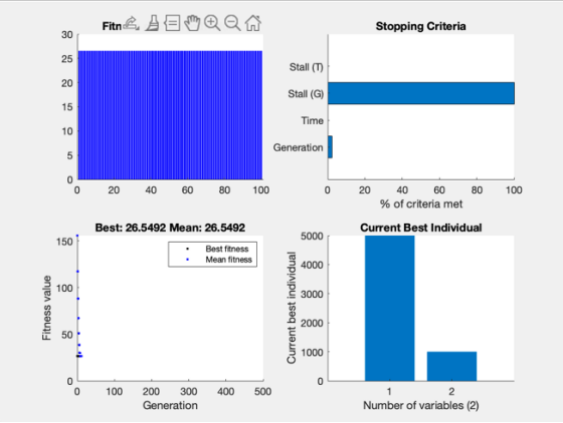


Figure 24 results of the generic algorithms for humidity and AC

3) Heart rate vs Emergence call

```
Single objective optimization:
2 Variable(s)

Options:
CreationFcn:       @gacreationuniform
CrossoverFcn:      @crossoverscattered
SelectionFcn:      @selectionroulette
MutationFcn:       @mutationadaptfeasible

                                 Best          Mean     Stall
Generation      Func-count        f(x)          f(x)    Generations
    1              200        5.585e+05     2.305e+06       0
    2              295        5.585e+05     1.626e+06       1
    3              390        5.585e+05     1.234e+06       2
    4              485        5.585e+05     9.343e+05       3
    5              580        5.585e+05     7.646e+05       4
    6              675        5.585e+05     6.852e+05       5
    7              770        5.585e+05     6.507e+05       6
    8              865        5.585e+05     6.044e+05       7
    9              960        5.585e+05     6.044e+05       8
   10             1055        5.585e+05     5.585e+05       9
   11             1150        5.585e+05     5.585e+05      10
```

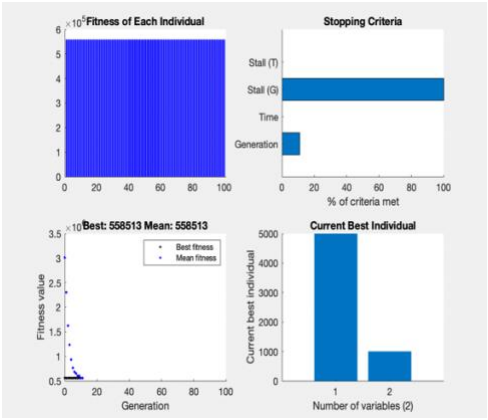


Figure 25 results of the generic algorithms for Heart rate and emergence call

The above are the outputs of the fitness function after performing optimization the other outputs are present In the appendix section.

V. Optimization on CEC'2005 Functions

In the rapid growth of technology and machine learning over the past decade, there have been several optimization algorithms proposed and tested few of the famous ones are the genetic algorithm and swarm optimization the list goes on. These techniques are used in real-world problems for optimization and are very good. With that said a comparison between algorithms

needs to be performed to find the difference. Although some algorithms can be good with a certain problem, not algorithms can suit all types of problems.

1) Shifted Rosebrook's Function

The equation of Shifted Rosebrook's function is given as follows:

$$F_6(x) = \sum_{i=1}^{D-1} \left(\left(100(z_i^2 - z_{i+1})\right)^2 + (z_i - 1)^2 \right) + f\_bias_6, z = x - 0 + 1, x = [x_1, x_2, \ldots, x_d]$$

D: dimensions
O = [o1,o2,…,on]

(10)

2) Shifted Rotated Griewank's Function

The equation is as follows:

$$F_7(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{z_i}{\sqrt{i}}) + 1 + f\_bias_7, z = (x - o) * M, x = [x_1, x_2, \ldots, x_D]$$

D: dimensions
O = [o1,o2,…,on]
M': linear transformation matrix, condition number = 3
M = M'(1 + 0.3|N(0,1)|)

(11)

The range for the 1 equation is from [-100 100] and the range for equation 2 is from [0, 600] we intend to change the D values and compare the outputs of the two algorithms the results are shown below.

A) Optimization outputs

The optimization algorithms were tested with two different D values. The results are shown below.

Genetic Algorithm :

CEC Function=7; D=2	
NO.	Local Minima
1	-179.9547
2	-179.9889
3	-179.9800
4	-179.9854
5	-179.9423
6	-179.992
7	-179.9777
8	-179.9998
9	-179.9867
10	-179.9564
11	-179.9684
12	-179.9924
13	-179.9900
14	-179.9886
15	-179.9423
Max_Value	-179.9423
Min_Value	-179.9998
Mean_Value	-179.9782
Standard_Deviation	0.0174

CEC Function=7; D=10	
NO.	Local Minima
1	-179.8171
2	-179.7064
3	-179.1381
4	-179.5046
5	-179.4995
6	-179.7399
7	-179.7399
8	-179.8386
9	-179.2705
10	-179.5732
11	-179.4850
12	-179.4014
13	-179.5166
14	-179.6246
15	-179.3553
Max_Value	-179.1381
Min_Value	-179.8563
Mean_Value	-179.5551
Standard_Deviation	0.2125

Table 16 outputs for Shifted Rosenbrock's Function

CEC Function=9; D=2	
NO.	Local Minima
1	-329.0366
2	-325.8570
3	-326.0057
4	-327.8287
5	-328.8394
6	-328.6637
7	-327.9624
8	-327.1840
9	-328.8262
10	-328.9856
11	-329.8882
12	-329.9026
13	-329.2317
14	-329.6245
15	-328.7877
Max_Value	-325.8570
Min_Value	-329.9026
Mean_Value	-328.4416
Standard_Deviation	1.2586

CEC Function=9; D=10.	
NO.	Local Minima
1	-326.9722
2	-320.4505
3	-324.1082
4	-325.1082
5	-326.0291
6	-325.9675
7	-317.9141
8	-320.5675
9	-324.7165
10	-325.3734
11	-324.6423
12	-320.3834
13	-323.5611
14	-328.4892
15	-323.5957
Max_Value	-317.9141
Min_Value	-328.4892
Mean_Value	-323.8736
Standard_Deviation	2.8809

Figure 17 Shifted Rotated Griewank's Function's Result

The principle behind the particle swamp is when a particle moves around according to the search space. The moment of every particle is decided by the local host. We have used two D values and two different function values. The experiment outputs are shown below.

Table 17 Shifter Rotated Griewank's Function

CEC Function = 7 , D = 2	
Max_value	-169.2807
Min_value	-169.2819
Mean_value	-169.2816
Standard_Deviation	0.00060747

Table 18 Shifter Rotated Griewank's Function

CEC Function = 7 , D = 10	
Max_value	-156.7260
Min_value	-157.4341
Mean_value	-157.2791
Standard_Deviation	0.1711

Table 19 Shifter Rotated Griewank's Function

CEC Function = 9 , D = 2	
Max_value	-329.0050
Min_value	-330.0000
Mean_value	-329.8010
Standard_Deviation	0.4120

Table 20 Shifter Rotated Griewank's Function

CEC Function = 9 , D = 10	
Max_value	-311.0958
Min_value	-328.0101
Mean_value	-321.0454
Standard_Deviation	5.5905

VI. Discussion

In this paper, we have created a fuzzy logic controller which can be used to automate an assistive care home for disabled people. Although our fuzzy controller can perform the job, it needed some optimization to find the global minimum and the global maximum. We have utilized the most used optimization algorithm the genetic algorithm to optimize our fuzzy system and we have successfully optimized our fuzzy system.

It was also found that the optimization algorithm is very vital when it comes to optimizing any problem. With that said we have also compared the genetic algorithm and swam algorithm and compared the results, it was found that genetic algorithms performed better than the particle swamp algorithms. Although it is not practical to say one algorithm is better than the other. A certain algorithm can be better at solving a particular problem but not all of them.

Upon conducting this experiment, there were no legal or ethical constraints. This experiment only benefits humans and the people in need therefore conducting any experiment benefiting disabled people is welcomed by the UK government.

On comparing our paper with the state of the art it pales in comparison, but we have conducted and very useful experiment which can be further improved by building a fuzzy logic controller for an entire flat. Which can address two major problems one is global warming and help disabled people.

VII. Conclusion

In Conclusion, we have built a fuzzy logic controller and successfully optimized the fuzzy logic controller. We have also used GP to find the undertints in the data and predicted the data far away from the future. with that said we faced some difficulties building the fuzzy logic controller, but all the problems were addressed and built and FLC can solve a real-world problem.

VIII. References

(2022). Retrieved 26 August 2022, from https://uk.mathworks.com/products/fuzzy-logic.html

(2022). Retrieved 26 August 2022, from https://uk.mathworks.com/products/fuzzy-logic.html

(2022). Retrieved 26 August 2022, from https://www.youtube.com/watch?v=pR8RPypQEAA&t=184s

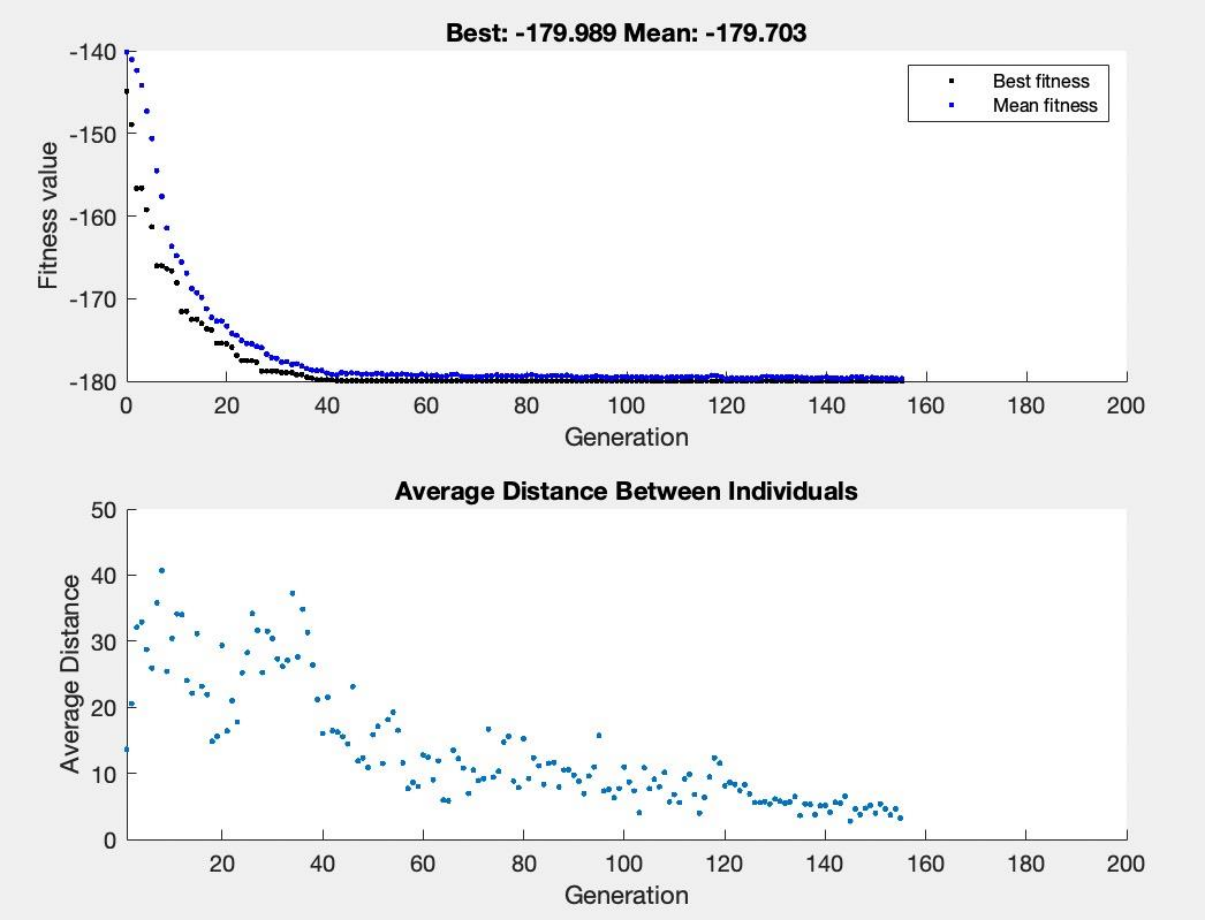
Levin, V., Yahya, A., & A. Boyarova, D. (2022). Predicting the technical condition of the power transformer using fuzzy logic and dissolved gas analysis method. *International Journal Of Electrical And Computer Engineering (ICE)*, *12*(2), 1139. doi: 10.11591/ijece.v12i2.pp1139-1146

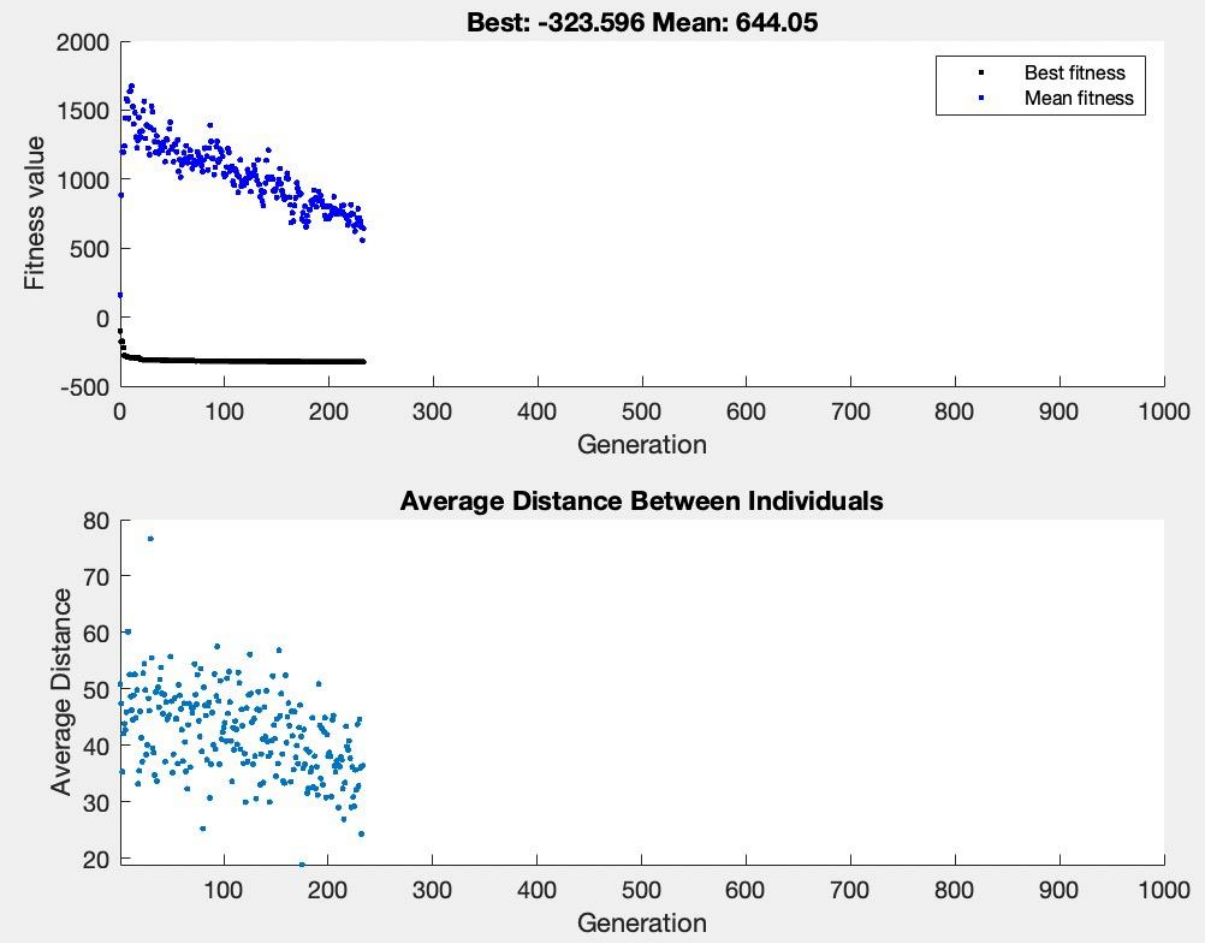
IX. Appendix

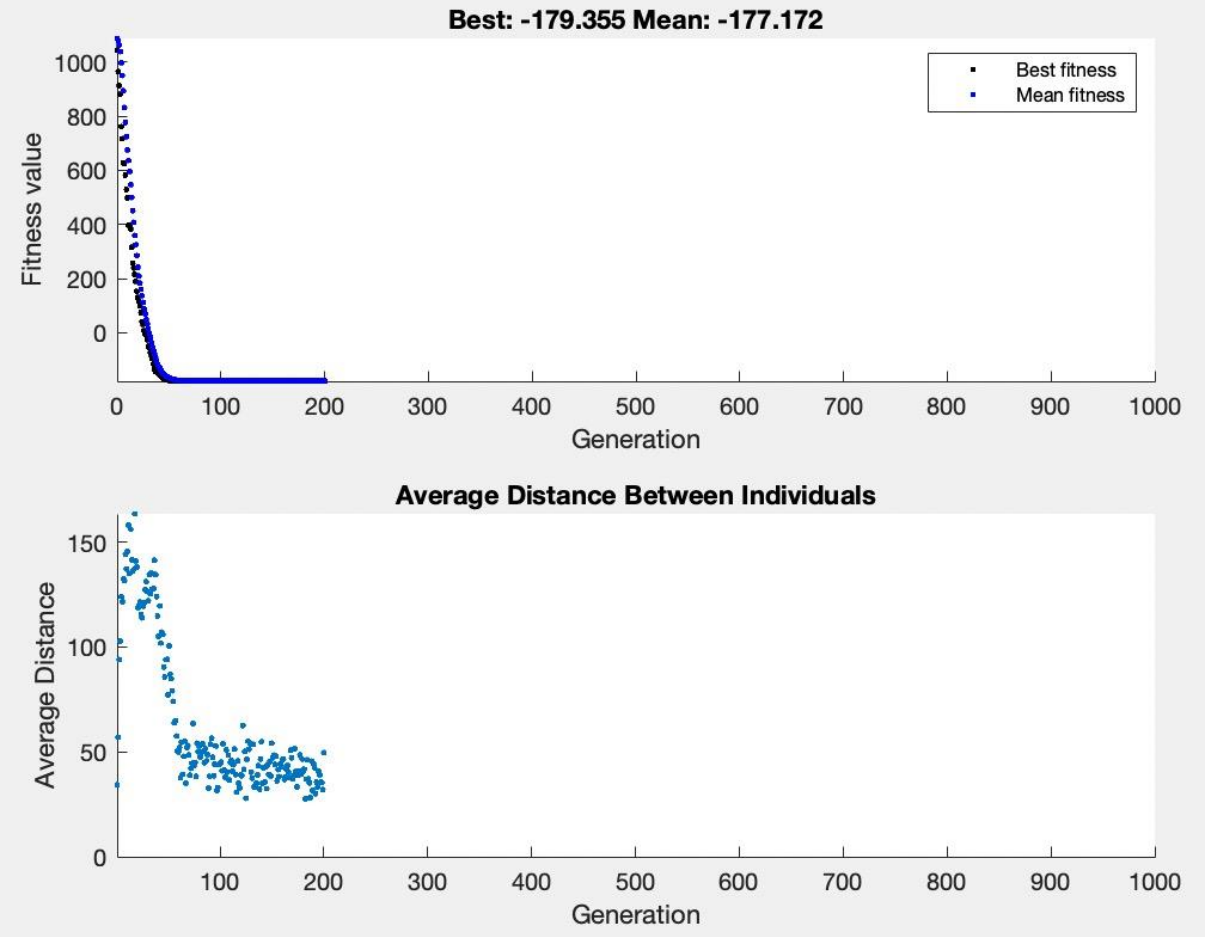
https://livecoventryac-my.sharepoint.com/:f:/g/personal/dhanasekar_uni_coventry_ac_uk/Eu6WL0bXxbdAp2JX6Spg1xMB-pSOPC514i4EehlDpR4OPg?e=CeIkWT

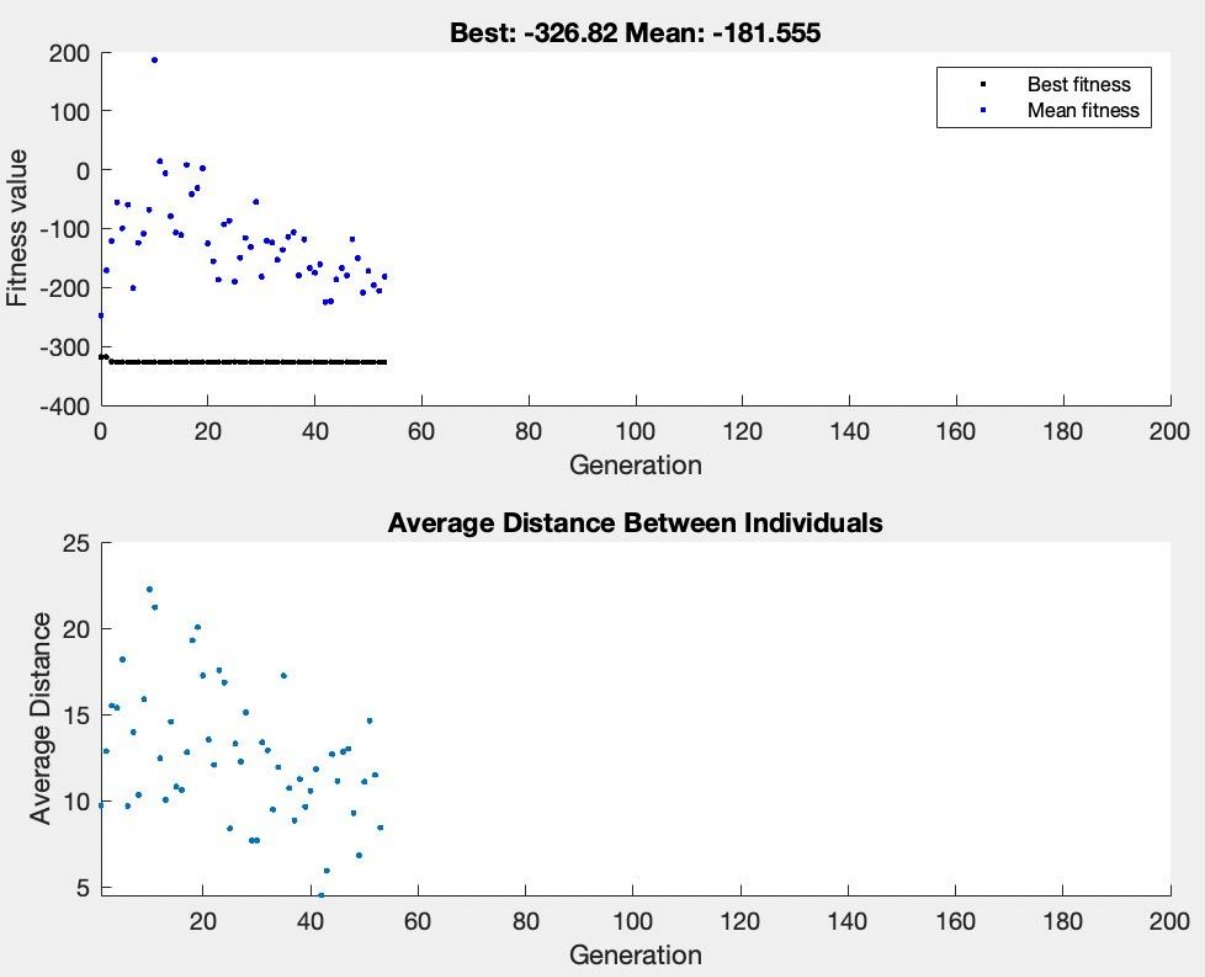
All the code involved in building the fuzzy logic controller and the files are uploaded in the above link

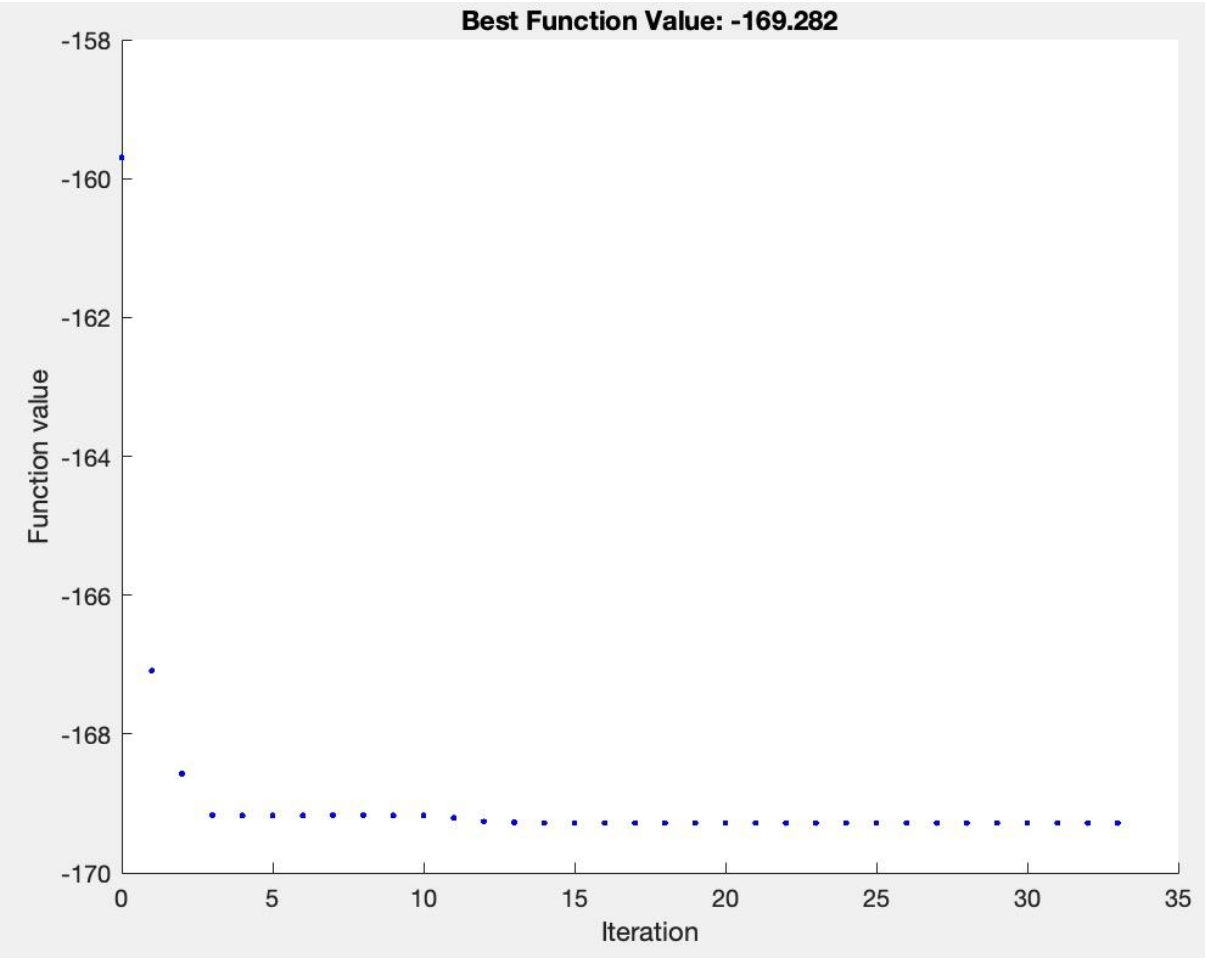
The table of contents was not added intentionally because it was too long and very shabby and doesn't make sense to make a table of contents for two separate reports.

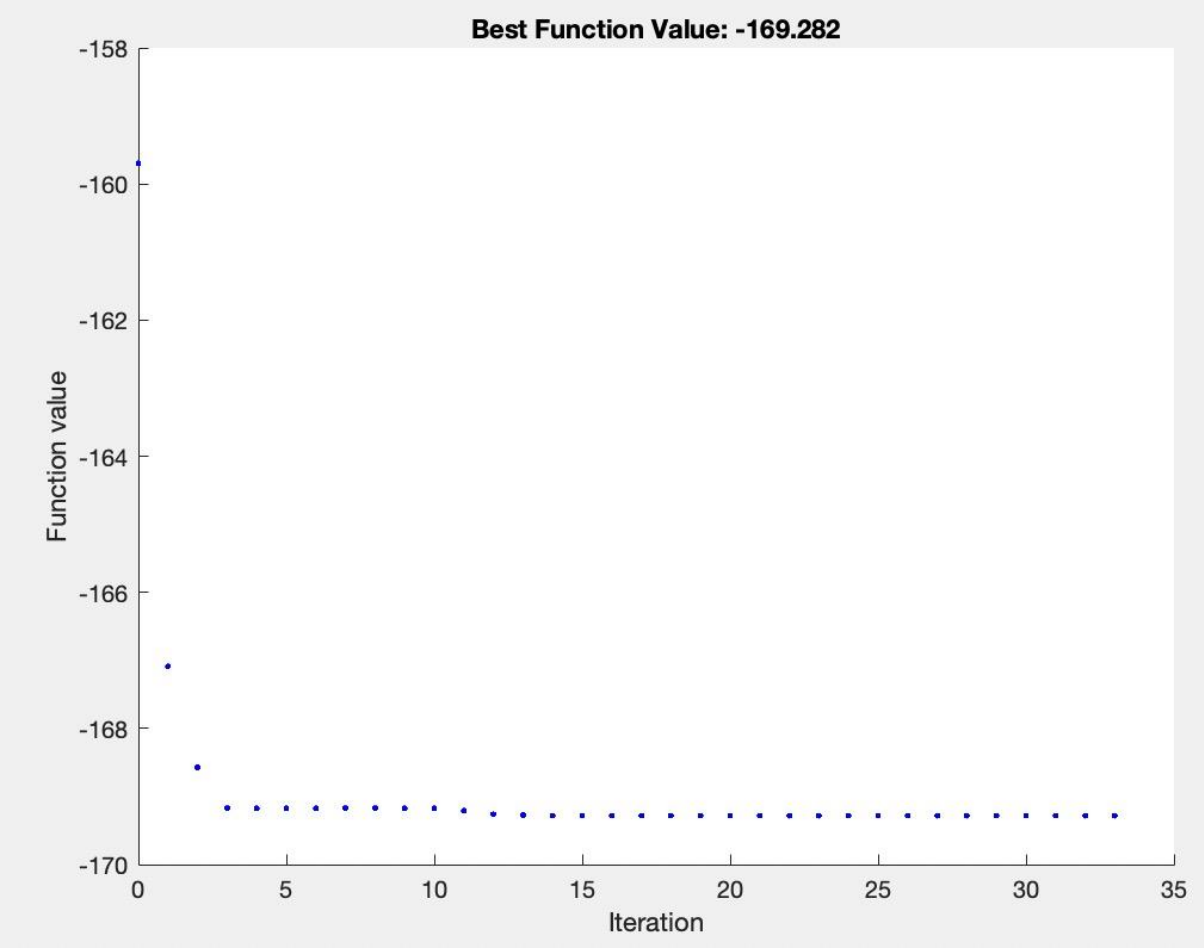

Best: -179.989 Mean: -179.703
Best fitness
Mean fitness
Fitness value
-140
-150
-160
-170
-180
0
20
40
60
80
100
120
140
160
180
200
Generation
Average Distance Between Individuals
Average Distance
50
40
30
20
10
0
20
40
60
80
100
120
140
160
180
200
Generation

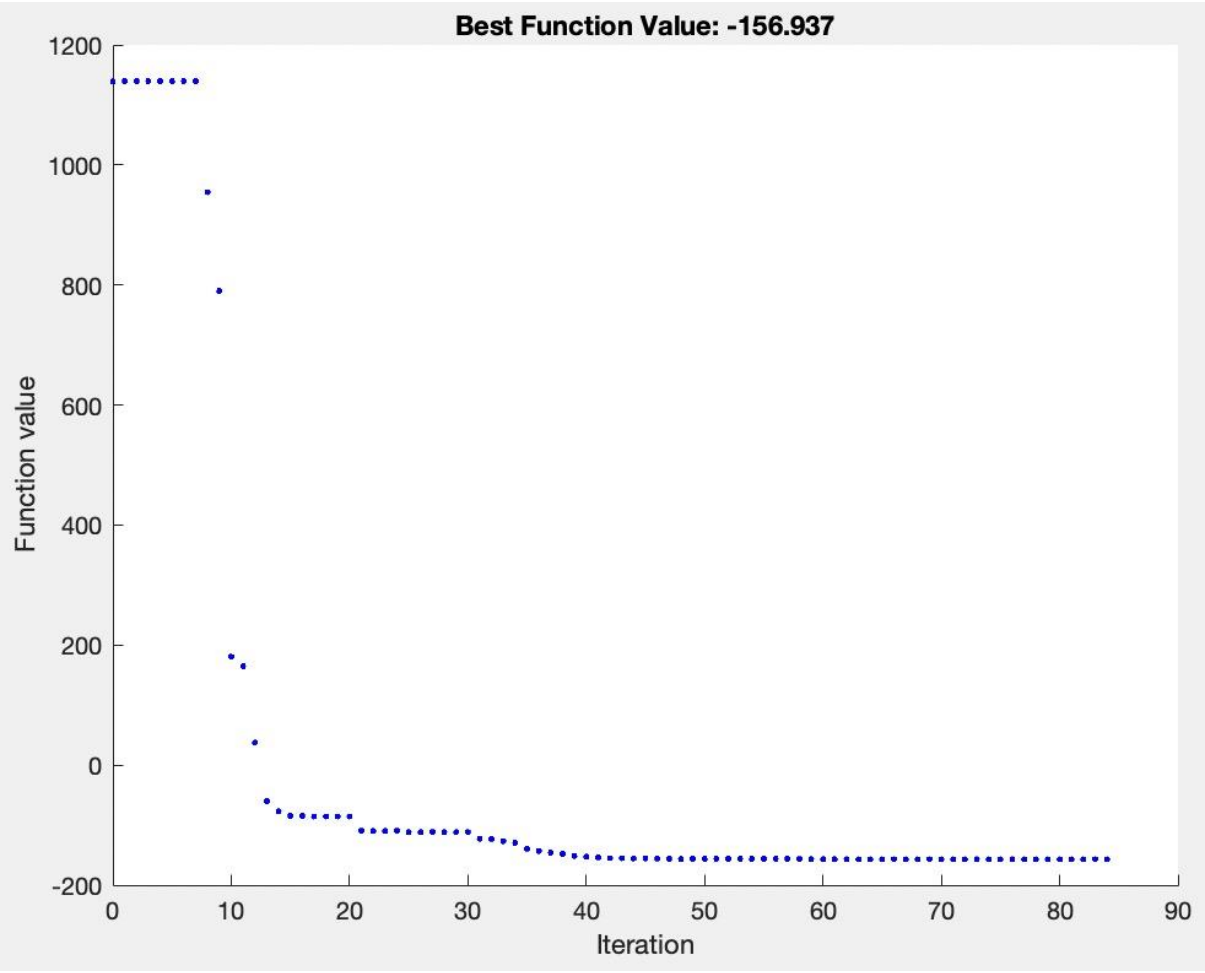

Best: -323.596 Mean: 644.05
Best fitness
Mean fitness
Fitness value
Generation
Average Distance Between Individuals
Average Distance
Generation

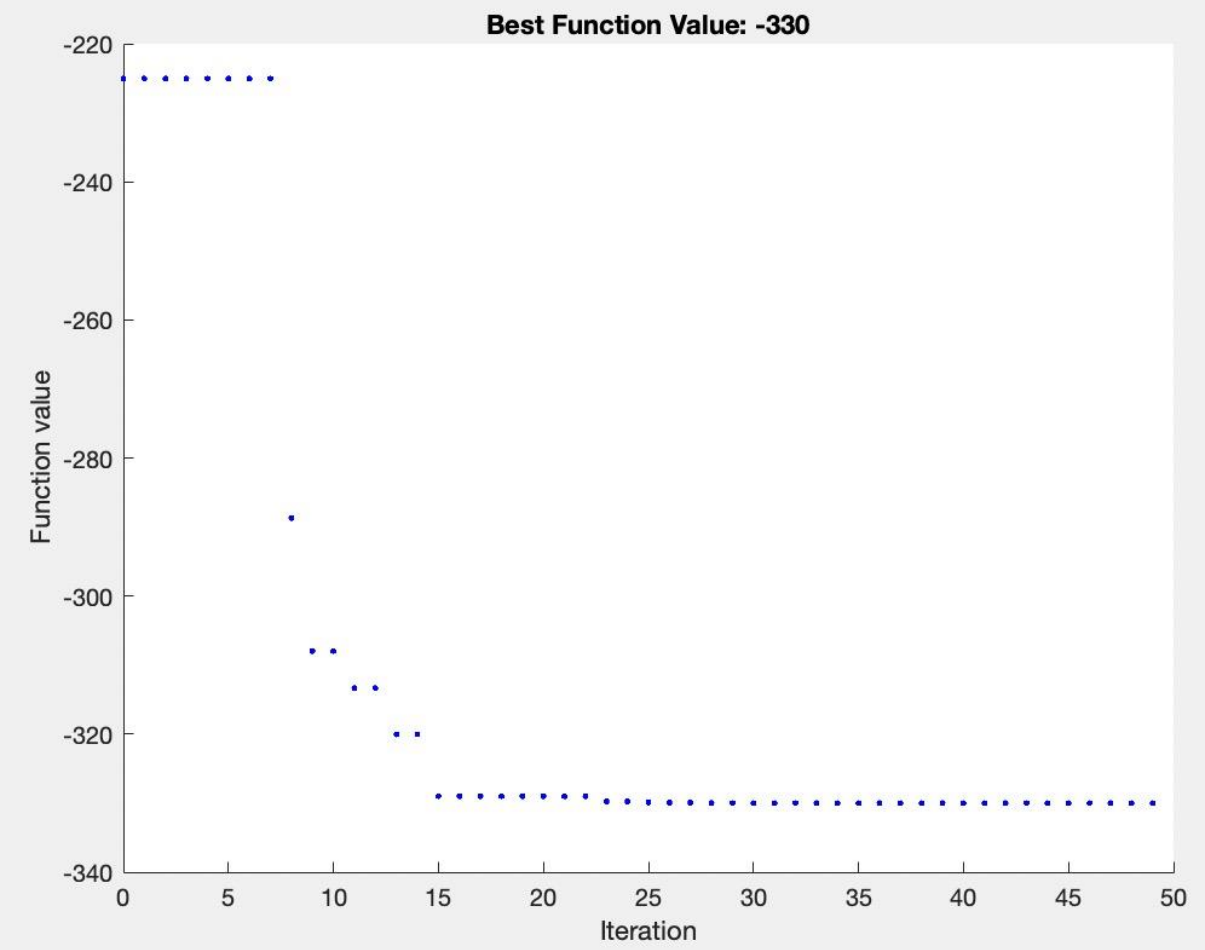

Best: -179.355 Mean: -177.172
Best fitness
Mean fitness
Fitness value
Generation
Average Distance Between Individuals
Average Distance
Generation


Best: -326.82 Mean: -181.555
Best fitness
Mean fitness
Fitness value
Generation
Average Distance Between Individuals
Average Distance
Generation


Best Function Value: -169.282
-158
-160
-162
-164
-166
-168
-170
Function value
0
5
10
15
20
25
30
35
Iteration


Best Function Value: -169.282
-158
-160
-162
-164
-166
-168
-170
Function value
0
5
10
15
20
25
30
35
Iteration


Best Function Value: -156.937
Function value
Iteration
1200
1000
800
600
400
200
0
-200
0
10
20
30
40
50
60
70
80
90


Best Function Value: -330
Function value
-220
-240
-260
-280
-300
-320
-340
0
5
10
15
20
25
30
35
40
45
50
Iteration

g