ECE9603 ASSIGNMENT 1

Tareq Tayeh - 250725776

OCTOBER 9, 2019

1. Description of the selected forecasted problem

Carbon Dioxide ($C0_2$) is a colorless greenhouse gas that absorbs and releases thermal radiation, creating the 'greenhouse effect'. CO_2 along with the rest of the greenhouse gases out there, are crucial for sustaining a habitable temperature on this planet. Without them, the earth would simply be too cold. However, since the industrial revolution and the massive increase in fossil fuel consumption, CO_2 emissions have skyrocketed. It has reached a stage where a climate change warning has been issued. It has significant ecological, physical, health, weather, agricultural and water impacts - the full list can be found in the 5th Intergovernmental Panel on Climate Change report [1].

To combat and try reducing all these impacts, the United Nations (UN) decided to set a target of limiting the average warming to 2^0C above the pre-industrial temperature [2]. As a good standing member of the UN, Canada has committed to decrease their CO_2 emissions over time, and in particular their CO_2 emissions per capita, as the country is ranked one of the worst 10 countries in the world in that category [3]. Canada's CO_2 emissions per capita takes into account the growing nation's population size. Understanding it give us a strong insight to how Canada is progressing and allows us to compare it with the progression of other countries.

Throughout the 'Our World in Data' website, Canada's CO_2 emissions per capita is available [3]. If this data can be used to quantitively forecast the future country's emissions per capita, Canada will be able to decide if they are on track to go below the specified threshold, or if additional regulations and actions need to be taken.

2. Description of available data (attributes, context, quantity…). Clearly indicate what attributes and/or parts you have used

The dataset obtained from the 'Our World in Data' website as mentioned in the previous section include 4 columns: 'Entity', 'Code', 'Year, and 'Per capita CO_2, emissions (tonnes per capita)'. The following table gives a description of each column and its description:

Column	Description	Data example
'Entity'	Contains the full names of all countries in the world.	'Canada', 'United Kingdom'
'Code'	Contains the code of the corresponding country in the 'Entity' column.	'CAN', 'GBR'
'Year'	Contains a single year in A.D. ranging from 1800 to 2017.	'1904', '2010'
'Per capita CO^2, emissions (tonnes per capita)'	Contains the CO^2 emissions per capita in tonnes per capita of the corresponding country in the 'Entity' column in the year specified under the corresponding 'Year' column	'9.02', '15.06'

Here is a snapshot of how the data looks like in the csv file:

Entity	Code	Year	Per capita CO2, emissions (tonnes per capita)
Canada	CAN	1800	0.005675991
Canada	CAN	1801	0.005609875
Canada	CAN	1802	0.005544526
Canada	CAN	1803	0.005479936
Canada	CAN	1804	0.005416096
Canada	CAN	1805	0.005353008
Canada	CAN	1806	0.005290646
Canada	CAN	1807	0.005229019
Canada	CAN	1808	0.005168104

The data received does not have to be seasonally adjusted as the data is collected at the same time of year every year, according to the OWID based on the Global Carbon Project; Carbon Dioxide Information Analysis Centre (CDIAC); Gapminder and UN population estimates. As mentioned in the 'Per capita CO_2 emissions' section under the 'Our World in Data' website [3], the per capita carbon dioxide emissions were derived based on four databases:

1. Long historical estimates of annual carbon dioxide emissions from the Carbon Dioxide Information Analysis Centre (CDIAC) [4].
2. Production and consumption-based estimates of CO_2 emissions from the Global Carbon Project [5].
3. Long-run population estimates by country, sourced from Gapminder & UN Population estimates [6].
4. Long-run global population estimates from the HYDE Database [7].

Per capita emissions were calculated by diving total national CO_2 emissions per year by population estimates, and CO_2 data has been converted from tonnes of carbon to tonnes of carbon dioxide (CO_2) using a conversion factor of 3.664.

For the purpose of this assignment, I will forecast the CO_2 emissions per capita for Canada based on the historical 'per capita CO^2, emissions (tonnes per capita)' data and its corresponding 'year' data, for all entries of 'Canada' under 'Entity' and codes 'CAN' under 'Code'. For a more accurate prediction, I will only include the years 1850 onwards due to the significant world industry changes since then. Therefore, I will have 168 data entries to work with. Figure 1 shows a visualization of Canada's 'per capita CO^2, emissions (tonnes per capita)' with respect to the year. A trend could be observed, where there is small increase from 1850 to 1878 before gradually increasing a lot then decreasing, then increasing again to a record high point before decreasing again and so on. As mentioned earlier, there is no seasonality.

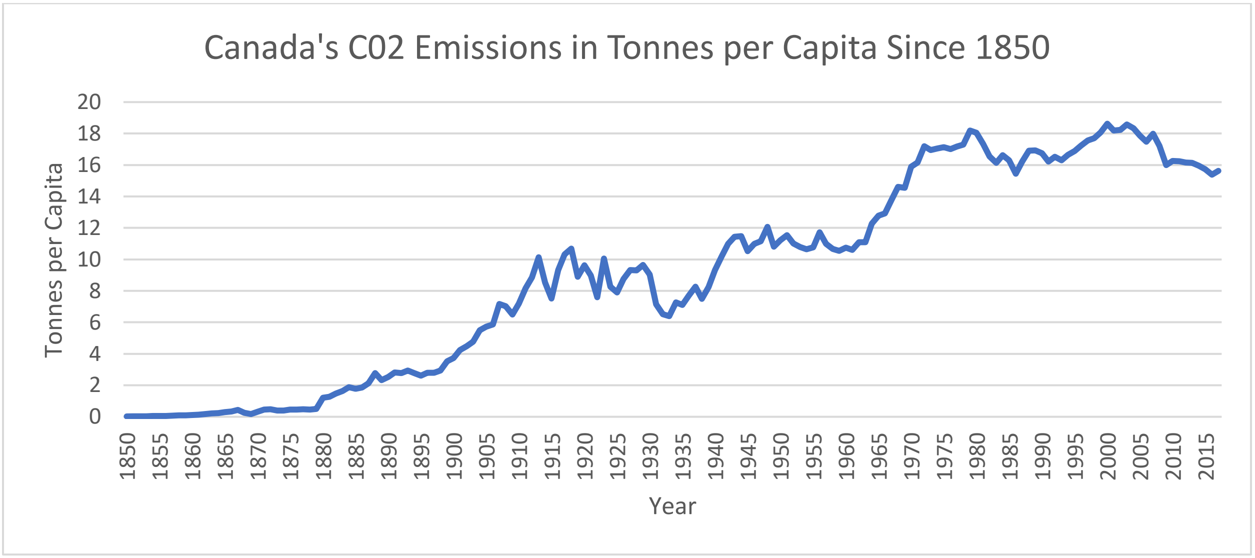
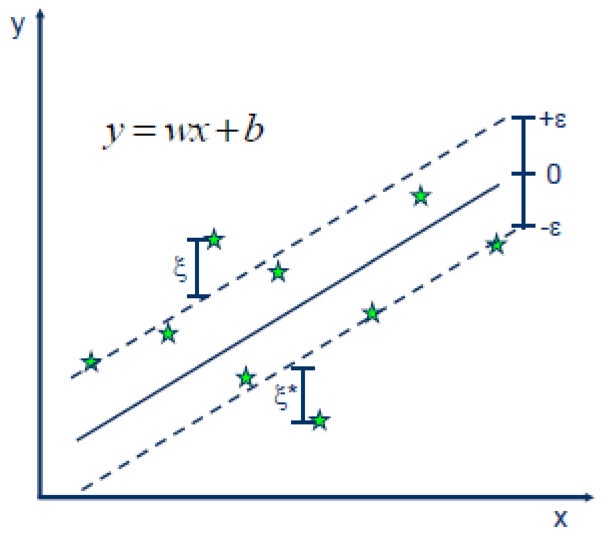


Figure 1

3. Short overview of the selected algorithms

(1) Support Vector Regression (SVR)

Support Vector Machines (SVM) is a popular supervised machine learning method used for classification and regression analysis. Its objective is to find the fattest hyperplane in an N-dimensional space with the largest margins possible that uniquely classifies data points, where N is the number of features. A margin is the distance separation between the hyperplane and the closest data points. If such a hyperplane exists, it is called the maximum-margin hyperplane. This allows for more 'wiggle-room' and less room for error. Since this deals with classification analysis really well, some deviations need to be applied for regression analysis. This is where Support Vector Regression (SVR) comes into place and excels. It uses the same principles as SVM; minimizing error and individualizing the hyperplane to maximize the margins. However, as the output is a real number, it becomes very difficult to predict due to the infinite possibilities, increasing the risk of error. To minimize this error, free parameter ε is introduced to serve as a threshold. All predictions must be within an ε range of the true predictions. Figure 2 shows an example of how this algorithm works.



• Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

• Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

Figure 2 [8]

(2) Gaussian Process Regression (GPR)

Gaussian processes (GP) are a powerful algorithm used for both regression and classification. Their main advantage is being able to give a reliable estimate based on their own uncertainty, due to their Bayesian nature of providing a probability distribution over possible functions. Gaussian Process Regression (GPR) is the GP-based technique used for regression analysis. Let us assume the linear function: *y = mx + b*. A prior distribution on the parameter m is specified and the probabilities based on the observed data are relocated. This now produces an updated distribution called the posterior distribution, which contains the information from the prior and observed data. The predictive distribution can be calculated for predictions of unseen points by weighing all the possible predictions of their calculated posterior distributions. That linear function for GPR now becomes *y = h(x)m + f(x)*, where *f(x)* are the latent variables and *h(x)* are the basis functions. This means the function keeps on updating with new points. Figure 3 shows a visualization of prior and posterior, and figure 4 shows an example of GPR in action for a given function via its prediction and confidence interval.

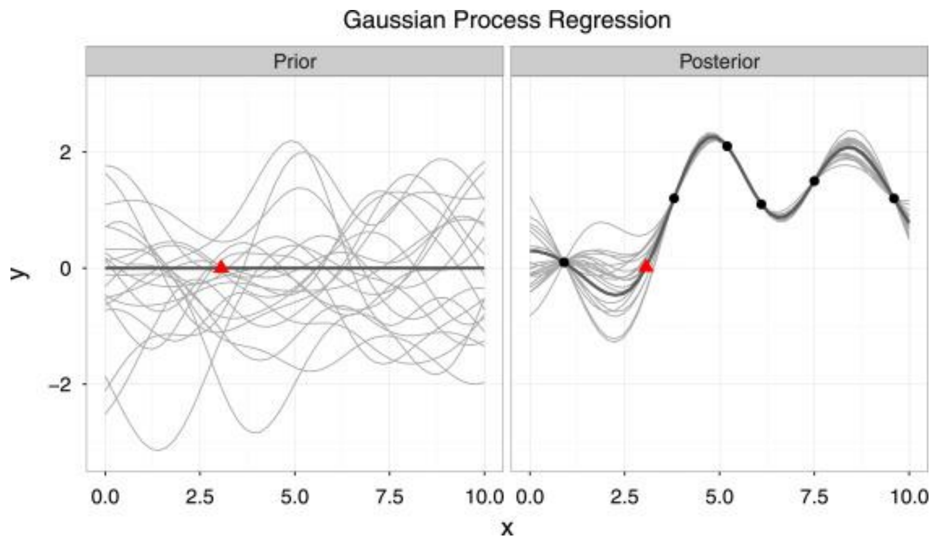


Figure 3 [9]

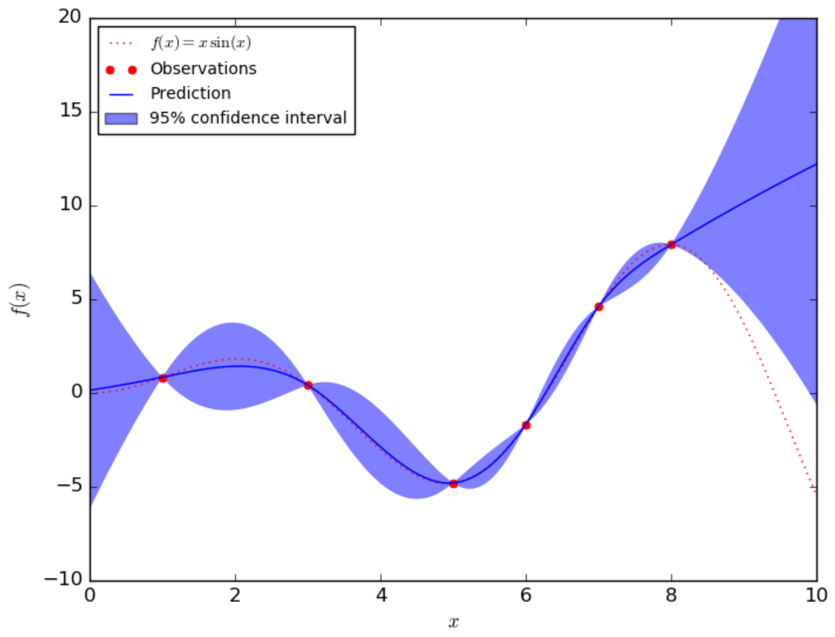


Figure 4 [10]

(3) Neural Network Regression (NNR)

An artificial neural network is a computing system inspired by the biological neural network. It consists of neurons that make simple mathematical decisions, and together, they can work on solving complex problems. The problems could consist of real-world data in images, sound, text, or time series, but they must all be translated numerically. A shallow neural network has three neuron layers: an input layer, a hidden layer, and an output layer. A deep neural network has an input layer and an output layer as well, but more than one hidden layers, adding complexity and prediction power to the model. Figure 5 visualizes this. Neural networks are usually used for clustering and classification, but they can pretend to be any type of regression model. Let us take the example illustrated in figure 6, which is utilizing a shallow neural network model. That is equivalent to a logistic regression with its error term, where it takes several dependent variables as input parameters, multiplies them by their coefficients as their weights, and runs them through the sigmoid activation and unit step functions.

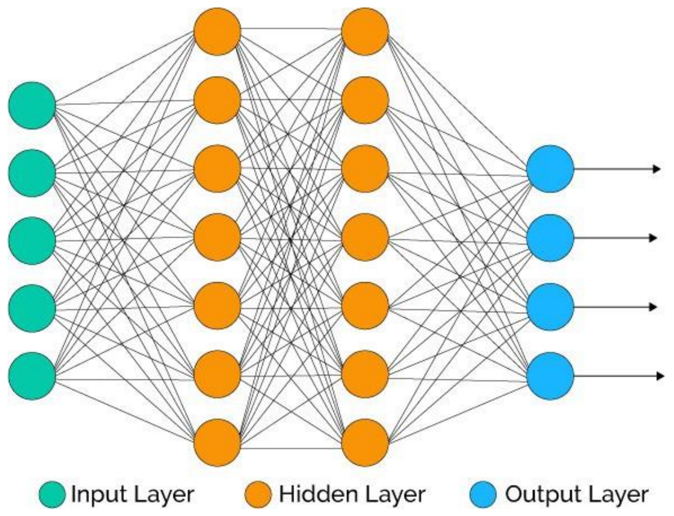


Figure 5 [11]

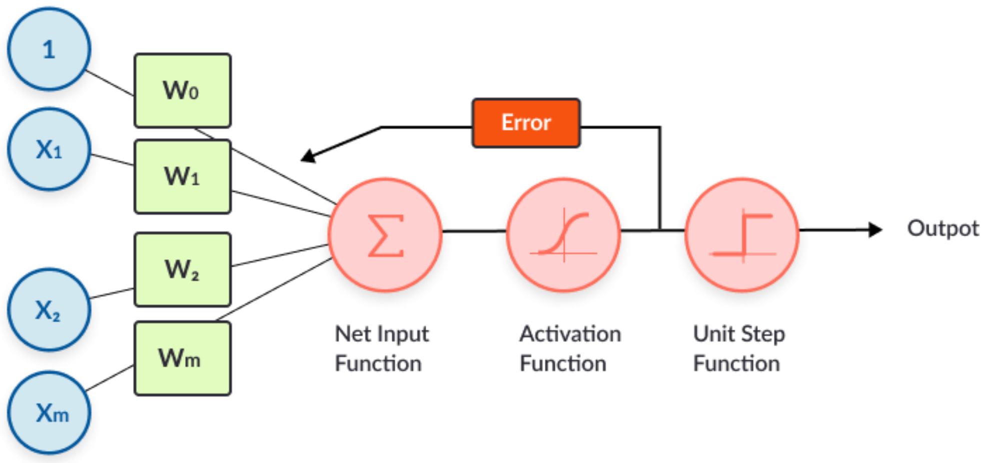


Figure 6 [12]

(4) Boosted Regression Tree Ensemble (BRTE)

Ensemble methods are algorithms that unite various algorithms into one predictive model in order to decrease bias and improve predictions. A regression tree ensemble, therefore, is a predictive model comprised of weighted combinations of various regression trees. These trees split the dataspace recursively and places a model within each split, where combining them increases the predictive performance. There are several classic methods to create ensemble methods, such as boosting, bagging, and random forests. Random forest contains a set of decision trees that are each trained on random subsets of the data. Boosting, however, builds trees one at a time, where each tree corrects the errors committed by the previous trees. I will be using the boosting technique in my assignment, making my implementation a Boosted Regression Tree Ensemble (BRTE). Figure 7 shows an example of a Simple Regression Tree Ensemble.

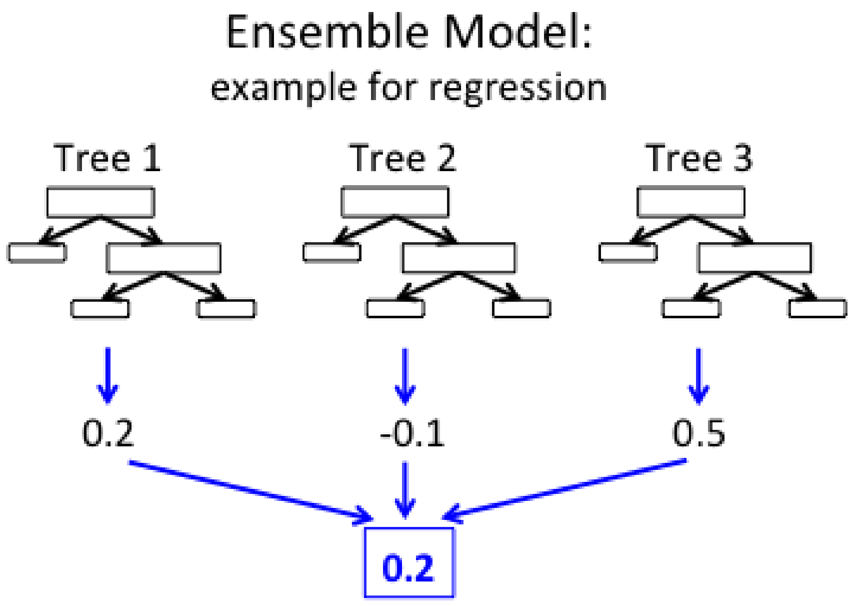


Figure 7 [13]

4. Specifics about how algorithms were applied and the evaluation procedure

All the algorithms specified above were implemented in Matlab. I know Dr.Grolinger mentioned not to use Matlab in the course, but as the code is not the main objective of this assignment nor is it graded, and I have been learning Machine Learning all through Matlab the past month, I opted to use it only for this assignment. Python will be used moving onwards in assignment 2 and the project.

As for the actual implementations, SVR, GPR & BRTE were all done through the command line utilizing Matlab's in-built functions. As this is a time-series forecasting method, the hold-out method was utilized as opposed to the 5-fold cross validation to test the validity of the different algorithms. It wouldn't make much sense using data in non-chronological form for testing here, as we're trying to predict the future and we would like to test how good our algorithms are in predicting those unseen/untrained test data. Therefore, the first 80% of the data were used for training the models, and the last 20% were used for testing the model (data is in year chronological order). No data standardization or parameter tuning/optimization were performed.

As for the NNR, I only found out after reaching to this section that building a 'future' forecasting model using a Neural Network on Matlab is much more complex than I thought, and it doesn't work as well with the set of data I got. But instead of ignoring this algorithm, I decided to use the in-built Matlab GUI Neural Net Time Series app for forecasting. I was sure I will be gaining some useful insight out of it. The application randomly selects data for training, validation, and testing though. I decided to use the 80% training – 10% validation – 10% testing split to evaluate the model's accuracy.

SVR – GPR – BRTE Implementation Steps

1. Import data from spreadsheet into a table and format
2. Split the data into training and testing
3. Fit a model using the training data and the parameter of interest from the spreadsheet
    - SVR fitting function: fitrsvm
      ```
      Mdl = fitrsvm(training,'Canada CO2 Emissions per Capita');
      ```
    - GPR fitting function: fitrgp
      ```
      Mdl = fitrgp(training,'Canada CO2 Emissions per Capita');
      ```
    - BRTE fitting function: fitrensemble
      ```
      Mdl = fitrensemble(training,'Canada CO2 Emissions per Capita');
      ```
4. Predict the test set (the emissions based on years) using the model created
    - `svrPrediction = predict(Mdl, test);`
    - `gprPrediction = predict(Mdl, test);`
    - `brtePrediction = predict(Mdl, test);`
5. Plot each model's prediction/result on the same chart alongside the original test data
    - `plot(year, emissions);`

- `plot(year(trainingSz+1:trainingSz+testSz), svrPrediction, '-red');`
- `plot(year(trainingSz+1:trainingSz+testSz), gprPrediction, '-green');`
- `plot(year(trainingSz+1:trainingSz+testSz), brtePrediction, '-black');`

The following figure (8) visualizes my results.

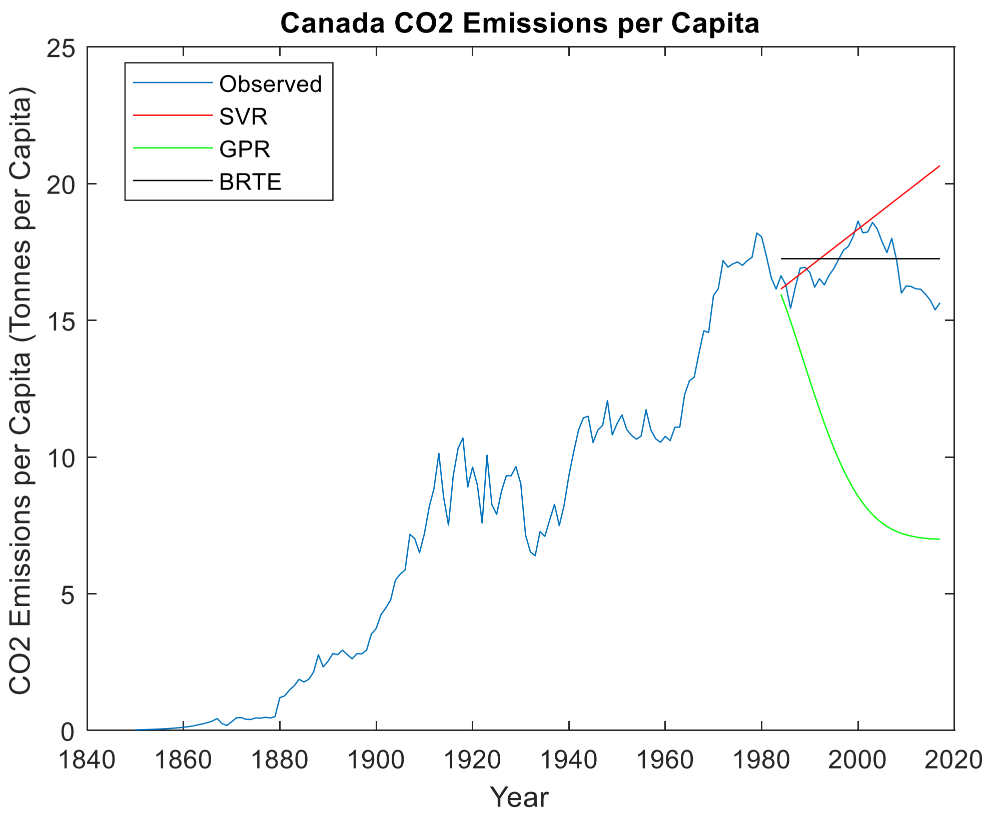
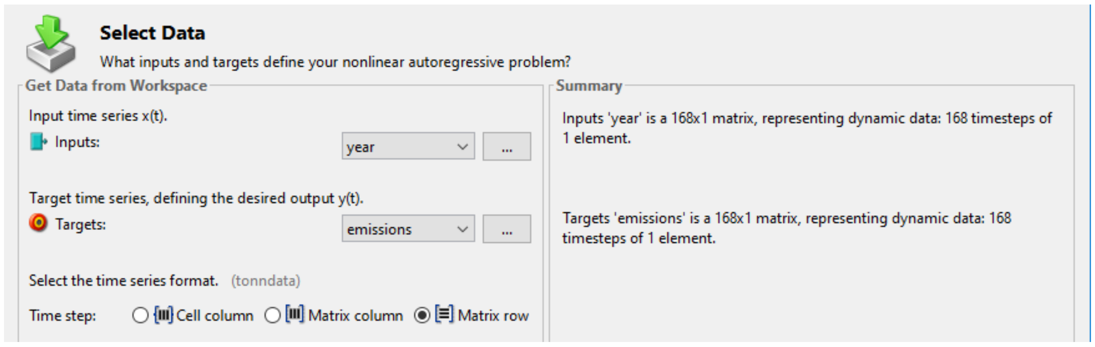


Figure 8

As it can be seen, BRTE was the best predictor, followed by SVM, followed by GPR. However, the optimistic self in me hopes that GPR's prediction is how the future of Canada's CO_2 emissions per capita is going to look like!

NNR Implementation Steps

1. Select Matlab’s Neural Net Time Series Application.
2. Select all 168 data entries of ‘year’ as inputs and ‘emissions’ as outputs.



Select Data

What inputs and targets define your nonlinear autoregressive problem?

Get Data from Workspace

Input time series x(t).

Inputs: year

Target time series, defining the desired output y(t).

Targets: emissions

Select the time series format. (tonndata)

Time step: ○ Cell column ○ Matrix column [✓ Matrix row]

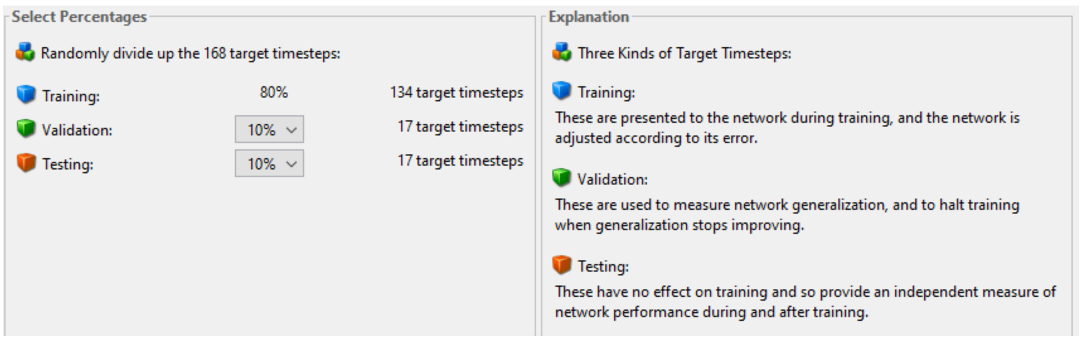
Summary

Inputs 'year' is a 168x1 matrix, representing dynamic data: 168 timesteps of 1 element.

Targets 'emissions' is a 168x1 matrix, representing dynamic data: 168 timesteps of 1 element.

Figure 9

3. Divide the data into 80% training, 10% validation, and 10% testing.



Select Percentages

Randomly divide up the 168 target timesteps:

Training:	80%	134 target timesteps
Validation:	10%	17 target timesteps
Testing:	10%	17 target timesteps

Explanation

Three Kinds of Target Timesteps:

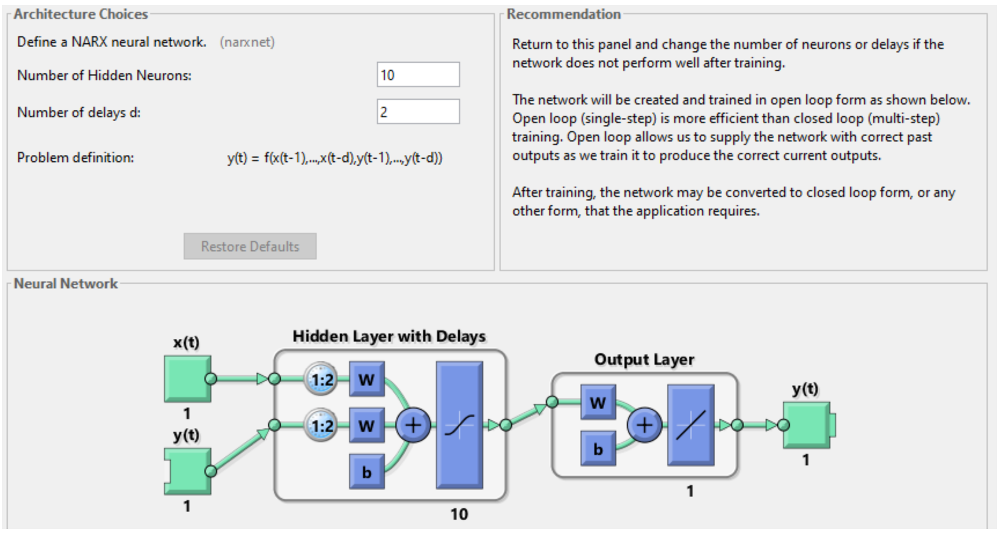
Training:
These are presented to the network during training, and the network is adjusted according to its error.

Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.

Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.

Figure 10

4. Select the # of hidden networks to be 10.



Architecture Choices

Define a NARX neural network. (narxnet)

Number of Hidden Neurons: 10

Number of delays d: 2

Problem definition: y(t) = f(x(t-1),...,x(t-d),y(t-1),...,y(t-d))

Recommendation

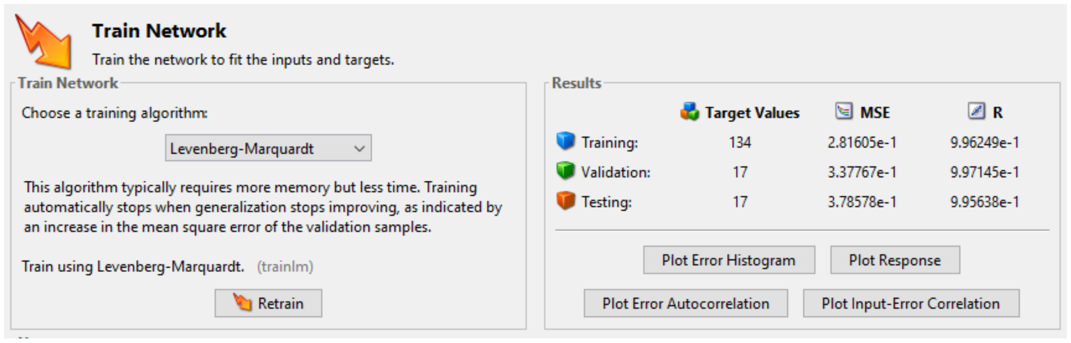
Return to this panel and change the number of neurons or delays if the network does not perform well after training.

The network will be created and trained in open loop form as shown below. Open loop (single-step) is more efficient than closed loop (multi-step) training. Open loop allows us to supply the network with correct past outputs as we train it to produce the correct current outputs.

After training, the network may be converted to closed loop form, or any other form, that the application requires.

Figure 11

5. Select the Levenberg-Marquardt training algorithm and train.


Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Train using Levenberg-Marquardt. (trainlm)

Retrain

Results

	Target Values	MSE	R
Training:	134	2.81605e-1	9.96249e-1
Validation:	17	3.37767e-1	9.97145e-1
Testing:	17	3.78578e-1	9.95638e-1

Plot Error Histogram | Plot Response | Plot Error Autocorrelation | Plot Input-Error Correlation

Figure 12

6. Plot response.

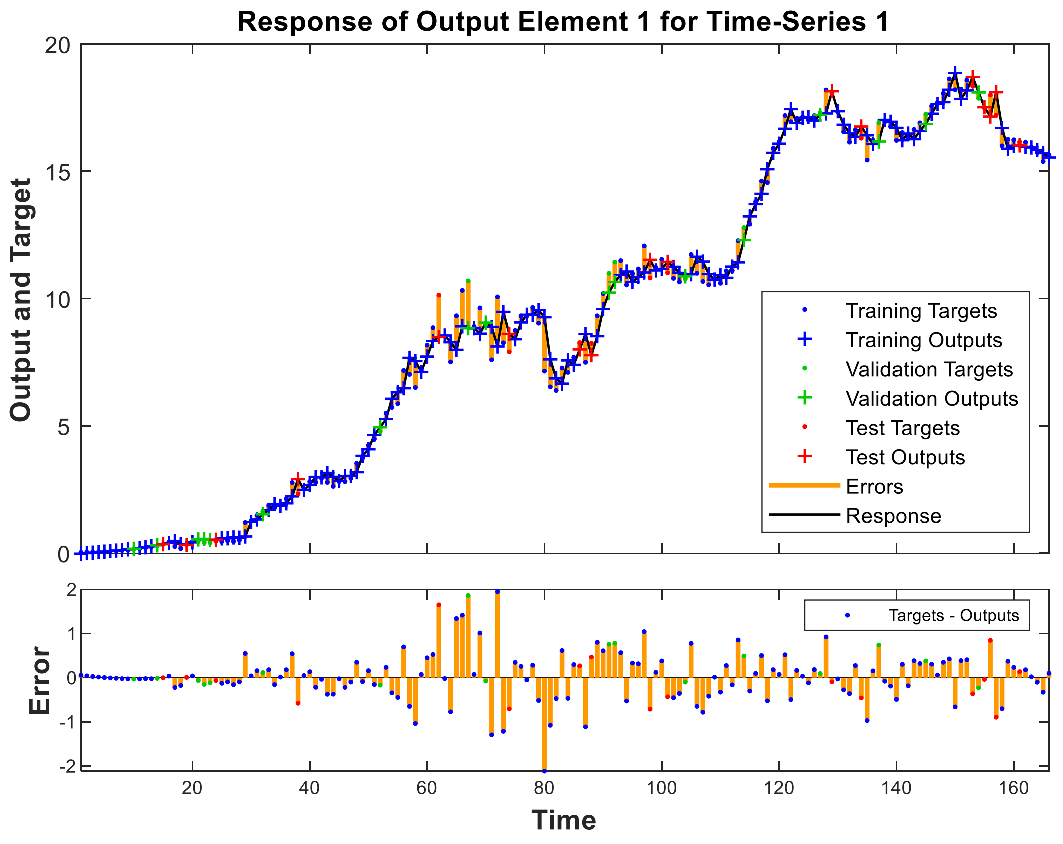


Figure 13

It can be observed that it predicted the random testing data without massive errors, which was expected.

5. Accuracy Comparison

Measuring forecast accuracy is not an easy task as there is no one size that fits all indicators. For this assignment, I will be utilizing the scale-dependant Root Means Square Error (RMSE) and Mean Absolute Percent Error (MAPE) to measure and compare the accuracy of all the regression models except the Neural Network Regression, due to its different implementation. These two very different indicators were utilized to overcome each other's limitation and offer a more general overview of the results.

RMSE

Defined as the square root of the average squared error or the Mean Squared Error (MSE). MSE is faster to compute and easier to manipulate than RMSE, however, RMSE does not treat each error the same, giving more importance to the biggest errors.

$$RMSE = \sqrt{\frac{1}{n}\sum e_t^2} = \sqrt{mean((y_i - \hat{y}_i)^2)}$$

Figure 14

```
%Calculating RMSE Errors in Matlab
svrRMSE=sqrt(mean((emissions((trainingSz+1):(trainingSz+testSz))-(svrPrediction)).^2));
gprRMSE=sqrt(mean((emissions((trainingSz+1):(trainingSz+testSz))-(gprPrediction)).^2));
brteRMSE=sqrt(mean((emissions((trainingSz+1):(trainingSz+testSz))-(brtePrediction)).^2));
```

MAPE

Defined as the sum of the individual absolute errors divided by the demand. In other words, the average of percentage errors. High errors during low demand periods will have significant impacts on MAPE.

$$MAPE = \frac{1}{n}\sum \frac{|e_t|}{d_t} = mean\left(\left|100 \cdot \frac{y_i - \hat{y}_i}{y_i}\right|\right)$$

Figure 15

```
%Calculating MAPE Errors in Matlab
svrMAPE = mean(abs(((svrPrediction - (test{:,2})) ./ (test{:,2})) * 100));
gprMAPE = mean(abs(((gprPrediction - (test{:,2})) ./ (test{:,2})) * 100));
brteMAPE = mean(abs(((brtePrediction - (test{:,2})) ./ (test{:,2})) * 100));
```

The following figures will reveal the RMSE and MAPE of each of the algorithms (specified in the previous section) predictive results.

Support Vector Regression

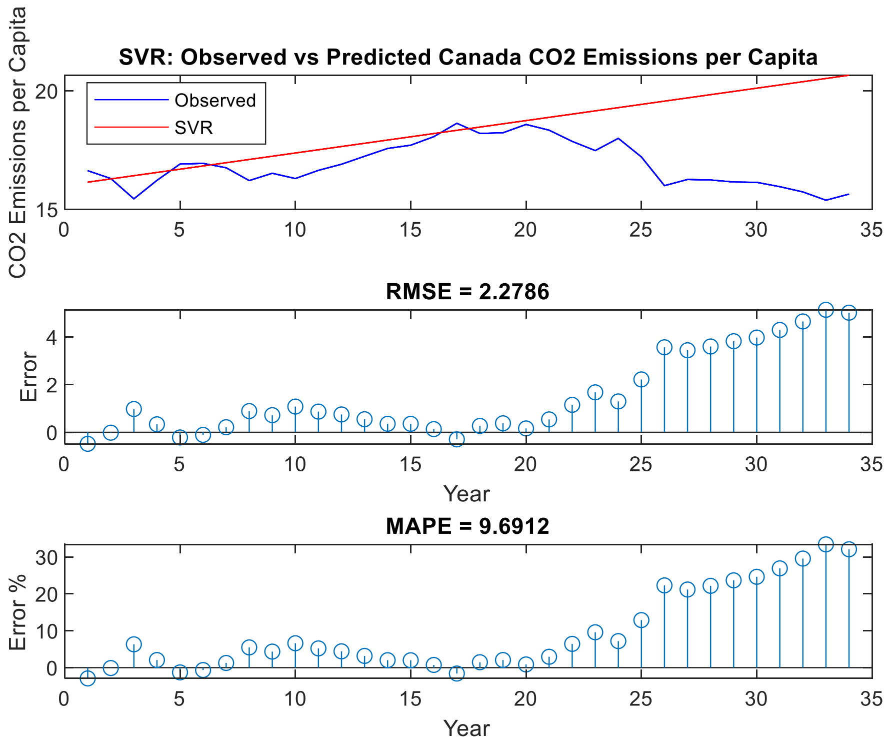


Figure 16

Gaussian Process Regression

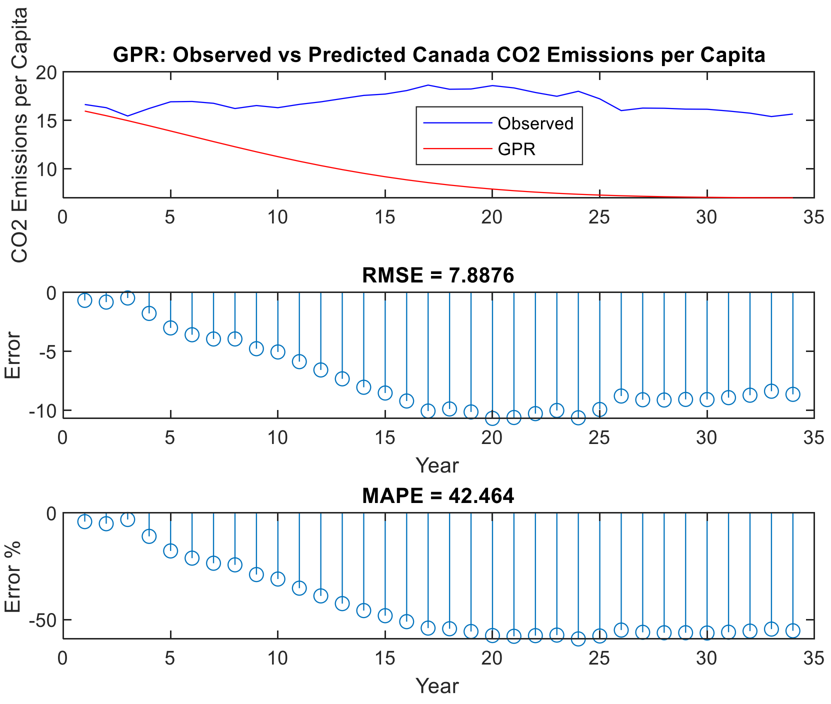


Figure 17

Boosted Regression Tree Ensemble

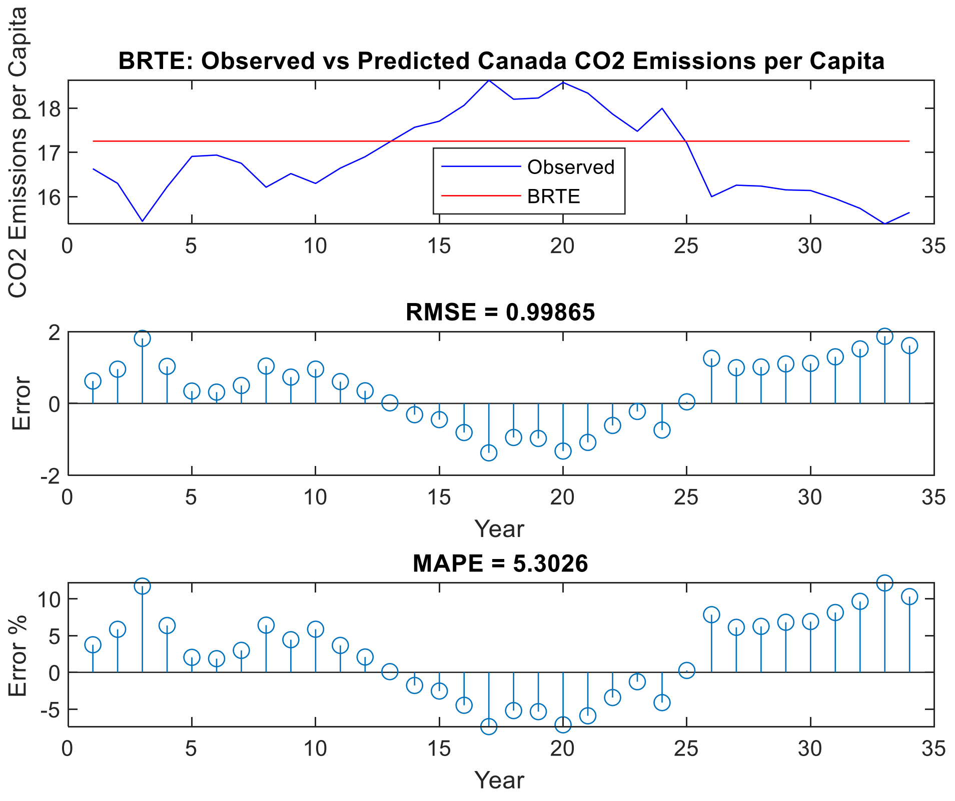
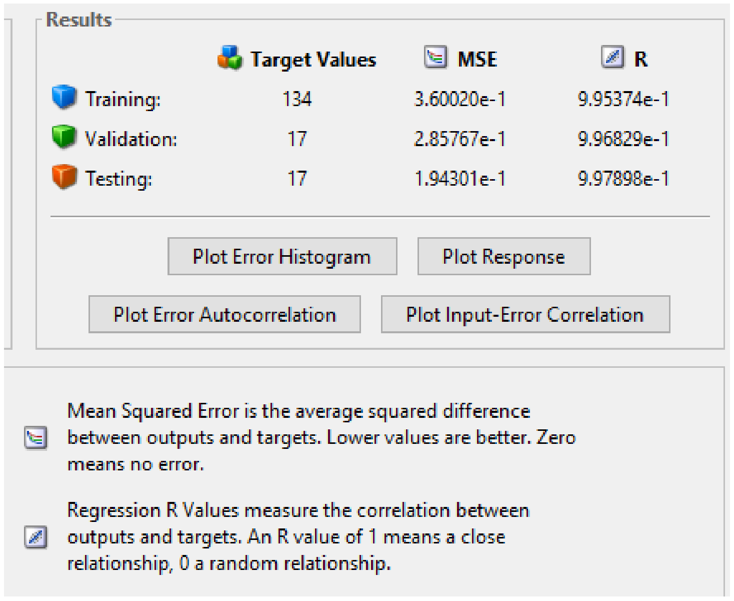


Figure 18

We can observe from the previous charts and graphs that the Boosted Regression Tree Ensemble proved to be the most effective technique with an RMSE value of 0.99865 and a MAPE value of 5.3%, followed by the Support Vector Regression technique which had an RMSE value of 2.2786 and a MAPE value of 9.6912%, followed by the Gaussian Process Regression which proved to be the least effective technique with an RMSE value of 7.8876 and a MAPE value of 42.464%.

Neural Network Regression

For this model, I will be utilizing the in-built Matlab GUI Neural Net Time Series app's error indicators. The following figure demonstrates the algorithm's Mean Squared Error (MSE) which is the RMSE squared, and the regression R values that measure the correlation between outputs and targets. We can see that both are significantly small indicating that the model had very little errors.



Results

	Target Values	MSE	R
Training:	134	3.60020e-1	9.95374e-1
Validation:	17	2.85767e-1	9.96829e-1
Testing:	17	1.94301e-1	9.97898e-1

Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

Figure 19

The following figure indicates the error histogram. Values are close to 0 indicating strong model accuracy.

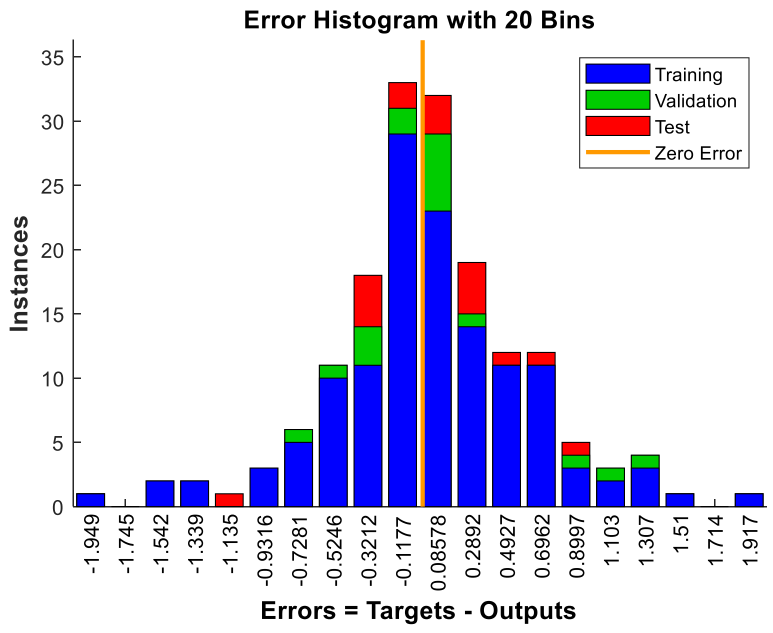


Figure 20

The following figure plots the error autocorrelation, where it's averaged as it's divided by the number of samples in the error data. The autocorrelation at lag 0 is equal to the MSE.

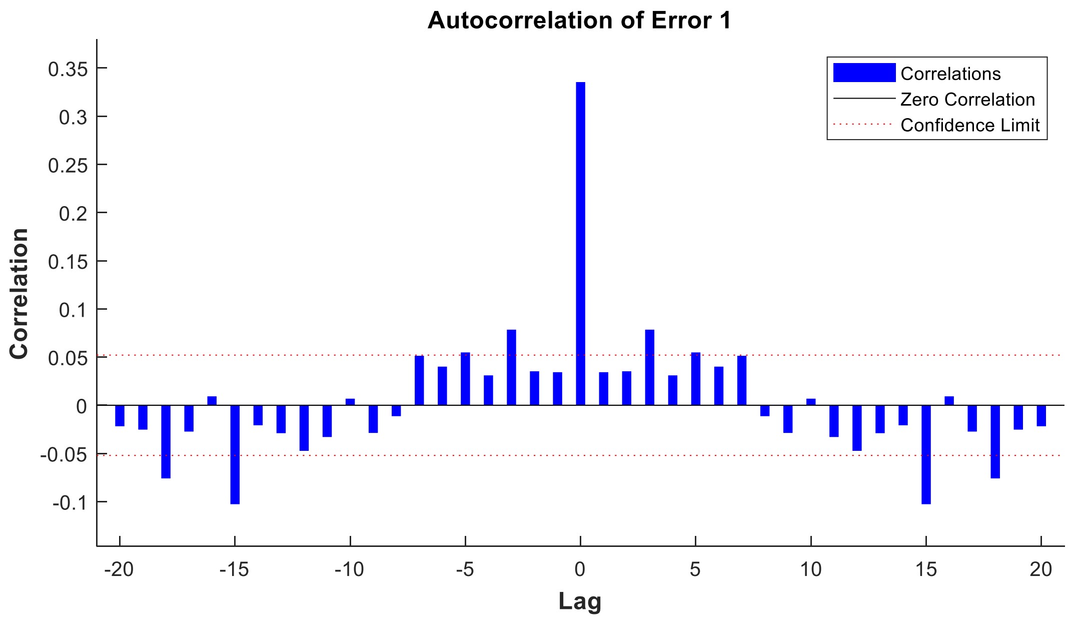


Figure 21

6. Code. Although there are no marks for the code itself, marks will be deducted if the code does not match the rest of the report.

```
%Import data to table & format
T = readtable('Canada_CO2_Emissions_per_Capita.xlsx','Range','C51:D219');
header = {'Year','Canada CO2 Emissions per Capita'};
T.Properties.VariableNames = header;

%Create and fill the variables required
year = T{:,1};
emissions = T{:,2};

%Determine the training and test size
trainingSz = round(0.80*size(T,1));
testSz = size(T,1) - trainingSz;

%Setting the training and test sets
training = head(T,trainingSz);
test = tail(T,testSz);

%Plot chart with 168 data points
figure(1)
plot(year, emissions);
title("Canada CO2 Emissions per Capita")
xlabel("Year")
ylabel("CO2 Emissions per Capita (Tonnes per Capita)")
hold on;

%SVR Model & Visualization
Mdl = fitrsvm(training,'Canada CO2 Emissions per Capita');
svrPrediction = predict(Mdl, test);
plot(year(trainingSz + 1 : trainingSz + testSz), svrPrediction, '-red');
hold on;

%GPR Model & Visualization
Mdl = fitrgp(training,'Canada CO2 Emissions per Capita');
gprPrediction = predict(Mdl, test);
plot(year(trainingSz + 1 : trainingSz + testSz), gprPrediction, '-green');
hold on;

%BRTE Model & Visualization
Mdl = fitrensemble(training,'Canada CO2 Emissions per Capita');
brtePrediction = predict(Mdl, test);
plot(year(trainingSz + 1 : trainingSz + testSz), brtePrediction, '-black');

%Add Legend to Chart
legend(["Observed","SVR","GPR", "BRTE"])

%Calculating RMSE Errors
svrRMSE = sqrt(mean((emissions((trainingSz + 1) : (trainingSz + testSz)) -
(svrPrediction)) .^2));
gprRMSE = sqrt(mean((emissions((trainingSz + 1) : (trainingSz + testSz)) -
(gprPrediction)) .^2));
```

```
brteRMSE = sqrt(mean((emissions((trainingSz + 1) : (trainingSz + testSz)) -
(brtePrediction)) .^2));

%Calculating MAPE Errors
svrMAPE = mean(abs(((svrPrediction - (test{:,2})) ./ (test{:,2})) * 100));
gprMAPE = mean(abs(((gprPrediction - (test{:,2})) ./ (test{:,2})) * 100));
brteMAPE = mean(abs(((brtePrediction - (test{:,2})) ./ (test{:,2})) * 100));

%Plot SVR Observed & Predicted & Errors
hold off;
figure(2);
subplot(3,1,1);
plot(test{:,2}, '-blue');
hold on;
plot(svrPrediction, '-red');
legend(["Observed","SVR"])
title("SVR: Observed vs Predicted Canada CO2 Emissions per Capita")
ylabel("CO2 Emissions per Capita")
subplot(3,1,2);
stem(svrPrediction - test{:,2});
title("RMSE = " + svrRMSE);
xlabel("Year");
ylabel("Error");
subplot(3,1,3)
stem(((svrPrediction - (test{:,2}))./(test{:,2}))*100);
title("MAPE = " + svrMAPE);
xlabel("Year");
ylabel("Error %");

%Plot GPR Observed & Predicted & Errors
figure(3);
subplot(3,1,1);
plot(test{:,2}, '-blue');
hold on;
plot(gprPrediction, '-red');
legend(["Observed","GPR"])
title("GPR: Observed vs Predicted Canada CO2 Emissions per Capita")
ylabel("CO2 Emissions per Capita")
subplot(3,1,2);
stem(gprPrediction - test{:,2});
title("RMSE = " + gprRMSE);
xlabel("Year");
ylabel("Error");
subplot(3,1,3)
stem(((gprPrediction - (test{:,2}))./(test{:,2}))*100);
title("MAPE = " + gprMAPE);
xlabel("Year");
ylabel("Error %");

%Plot BRTE Observed & Predicted & Errors
figure(4);
subplot(3,1,1);
plot(test{:,2}, '-blue');
hold on;
plot(brtePrediction, '-red');
legend(["Observed","BRTE"])
title("BRTE: Observed vs Predicted Canada CO2 Emissions per Capita")
```

```
ylabel("CO2 Emissions per Capita")
subplot(3,1,2);
stem(brtePrediction - test{:,2});
title("RMSE = " + brteRMSE);
xlabel("Year");
ylabel("Error");
subplot(3,1,3)
stem(((brtePrediction - (test{:,2}))./(test{:,2}))*100);
title("MAPE = " + brteMAPE);
xlabel("Year");
ylabel("Error %");
```

References

[1] *AR5 Climate Change 2014: Impacts, Adaptation, and Vulnerability — IPCC*. (2019). *Ipcc.ch*. Retrieved 6 October 2019, from https://www.ipcc.ch/report/ar5/wg2/

[2] *Raising Climate Ambition Essential for 2 Degrees C Goal - Economist Nick Stern | UNFCCC*. (2018). *Unfccc.int*. Retrieved 6 October 2019, from https://unfccc.int/news/raising-climate-ambition-essential-for-2-degrees-c-goal-economist-nicholas-stern

[3] Ritchie, H., & Roser, M. (2017). CO_2 and Greenhouse Gas Emissions. *Our World In Data*. Retrieved from https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions

[4] Laboratory, O. (2019). *Carbon Dioxide Information Analysis Center (CDIAC)*. *Cdiac.ess-dive.lbl.gov*. Retrieved 6 October 2019, from https://cdiac.ess-dive.lbl.gov/

[5] *Carbon Budget*. (2019). *Globalcarbonproject.org*. Retrieved 6 October 2019, from https://www.globalcarbonproject.org/carbonbudget/index.htm

[6] *Population*. (2019). *Our World in Data*. Retrieved 6 October 2019, from https://ourworldindata.org/grapher/population

[7] *Population - the Netherlands Environmental Assessment Agency (PBL)*. (2010). *Themasites.pbl.nl*. Retrieved 6 October 2019, from https://themasites.pbl.nl/tridion/en/themasites/hyde/basicdrivingfactors/population/index-2.html

[8] *Support Vector Regression*. (2019). *Saedsayad.com*. Retrieved 7 October 2019, from https://www.saedsayad.com/support_vector_machine_reg.htm

[9] Schulz, E., Speekenbrink, M., & Krause, A. (2018). A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal Of Mathematical Psychology*, *85*, 1-16. doi:10.1016/j.jmp.2018.03.001

[10] *1.7. Gaussian Processes — scikit-learn 0.17.1 documentation*. (2019). *Scikit-learn.org*. Retrieved 7 October 2019, from https://scikit-learn.org/0.17/modules/gaussian_process.html

[11] *When not to use Neural Networks*. (2018). *Medium*. Retrieved 7 October 2019, from https://medium.com/datadriveninvestor/when-not-to-use-neural-networks-89fb50622429

[12] *Neural Networks for Regression (Part 1)—Overkill or Opportunity? - MissingLink.ai*. (2019). *MissingLink.ai*. Retrieved 7 October 2019, from https://missinglink.ai/guides/neural-network-concepts/neural-networks-regression-part-1-overkill-opportunity/

[13] *Random Forests and Boosting in MLlib*. (2015). *Databricks*. Retrieved 8 October 2019, from https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html