The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Student ID: 20311375

Supervisor: Ms. Susan Ali

Module Code: COMP3003

2021/05

Support Vector Regression and Vector Autoregression for Predicting Energy Consumption in London: A Review and Comparison

Submitted May 2021, in partial fulfilment of

the conditions for the award of the degree BSc (Hons) Computer Science with Artificial Intelligence.

20311375

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature [illegible]

Date 10 / 05 / 2021

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

Abstract

The dissertation studies how vector autoregression (VAR) could be used to predict the energy consumption of households in London in the form of a multivariate time series forecast graph. Support vector regression (SVR), a machine learning model, will also be used to produce a univariate time series forecast graph in order to be compared with the widely used econometric model VAR. Based on the data collected, a time series forecast is produced in the hope of providing insight for energy suppliers, especially in London.

Acknowledgement

I would like to thank my supervisor, Ms. Susan Ali, for her advice and guidance throughout this project. I would also like to thank my colleague, Yoksan Herlie for his advice throughout this project. The main data collection of the smart meter has been done by the UK power networks. The big work of data compilation was done by Jean-Michel D. and was uploaded to Kaggle. The details related to the ACORN group were provided by CACI. The weather data are obtained from Dark Sky's API.

Table of Contents

1. Introduction

1.1. Motivation

Every year London spends around £3.9 billion on their gas and electricity bills, with the annual average energy bill of £1,17512 per household. London set a challenge as part of its 2011 Climate Change and Energy Strategy, to reduce the city's carbon dioxide emissions. The Mayor, in his 2016 Manifesto, pledged to commit London to becoming a zero-carbon city by 2050, and to establish "Energy for Londoners", an energy services company that would "help Londoners to generate more low-carbon energy and increase their energy efficiency". Despite its efforts, London is falling well behind on its milestones to 2025 and will need to step up its energy efficiency actions.

The aforementioned spending of £3.9 billion is money that does not stay in London's economy. Improving efficiency and cutting energy costs translates to more money being invested in and on London's economy.

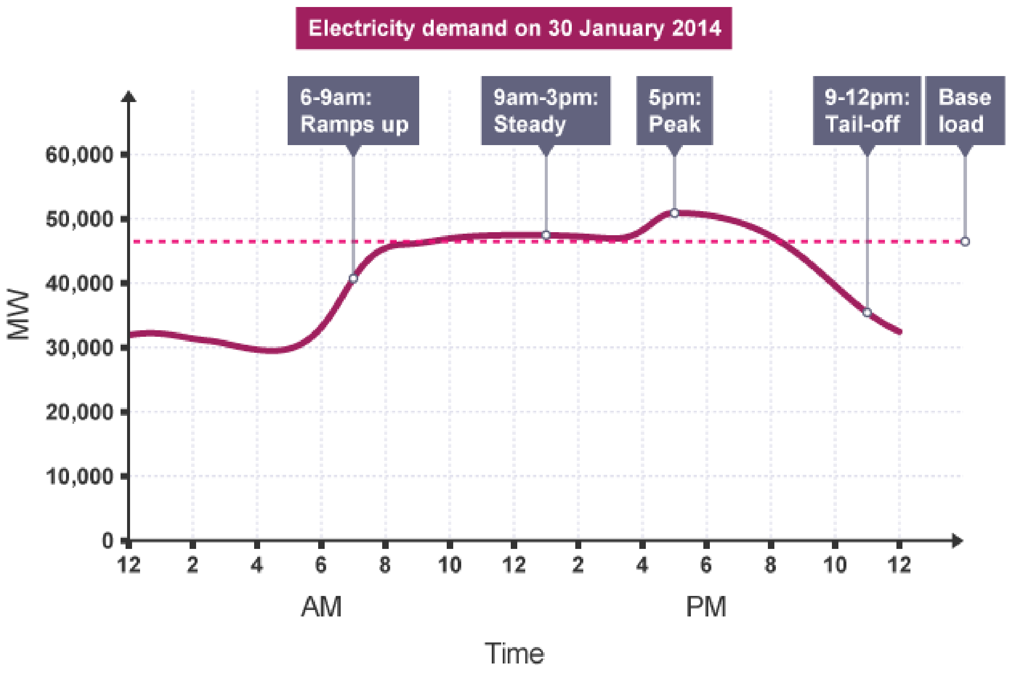


Fig. 1 - Variation of energy demand in a day

When the demand for electricity increases as shown in *Fig. 1*, the National Grid reacts by providing electricity. Nuclear and coal powered stations run all the time and produce the base load of electricity, this is due to their longer start up time. Hydroelectric power stations have a short start-up time and are used to provide additional electricity at peak times. In the event where the National Grid produces an excess of energy, sometimes grid energy storage is used. This can be in the form of a pumped hydro storage, flywheel energy storage or even a Li-ion battery farm. Storing excess energy still has it downsides, as it is not 100% efficient. Meaning that the energy output from the storage will be less than what was put in. Pumped hydro storage have an efficiency rating of 70-85%, flywheels have an efficiency rating of 70-95% and Li-ion battery farms have an efficiency rating of 85-95%. Ultimately, the goal is to be able to produce the exact amount of energy as requested at any given time. This could prove valuable for energy suppliers, as they could increase efficiency by providing just the right amount of energy at a certain time frame and also, they could more accurately predict when a service is due, for their infrastructure to keep up with future energy demand. At the moment that is near to impossible to predict with that much accuracy, however there are several ways we can try to get close to it. Time series data analysis is a fundamental part of any type of any business decision, as its goal is to make a forecast for the future.

1.2. Aims and Objectives

The main aim for this project is to create a program capable of forecasting energy consumption in London to be of significant use. The output of which will be in a form of a time series graph.

The key objectives for this project are:

a. Investigate existing industry solutions for predicting energy consumption.

b. Design and develop a support vector regression (SVR) model for a univariate time series forecast.

c. Design and develop a vector autoregression (VAR) model for a multivariate time series forecast.

d. Create visualizations to provide a deeper understanding on the forecast, and to generally have a more readable output.

e. Compare and evaluate the two models (SVR and VAR).

2. Related Work

On the 20th of October 2017, Jean-Michel D made an article explaining his approach to creating a forecast system of energy consumption in metropolitan France. In his article, he implemented a Power Temperature Gradient (PTG) technique to forecast energy consumption. It shows a linear regression model on how temperature change affects energy consumption, as shown in *Fig. 2*. The model is a piecewise regression with a winter part represented by the linear regression (negative gradient), and a summer part with the constant line (zero gradient). The aforementioned model considers heating needs and appliances.

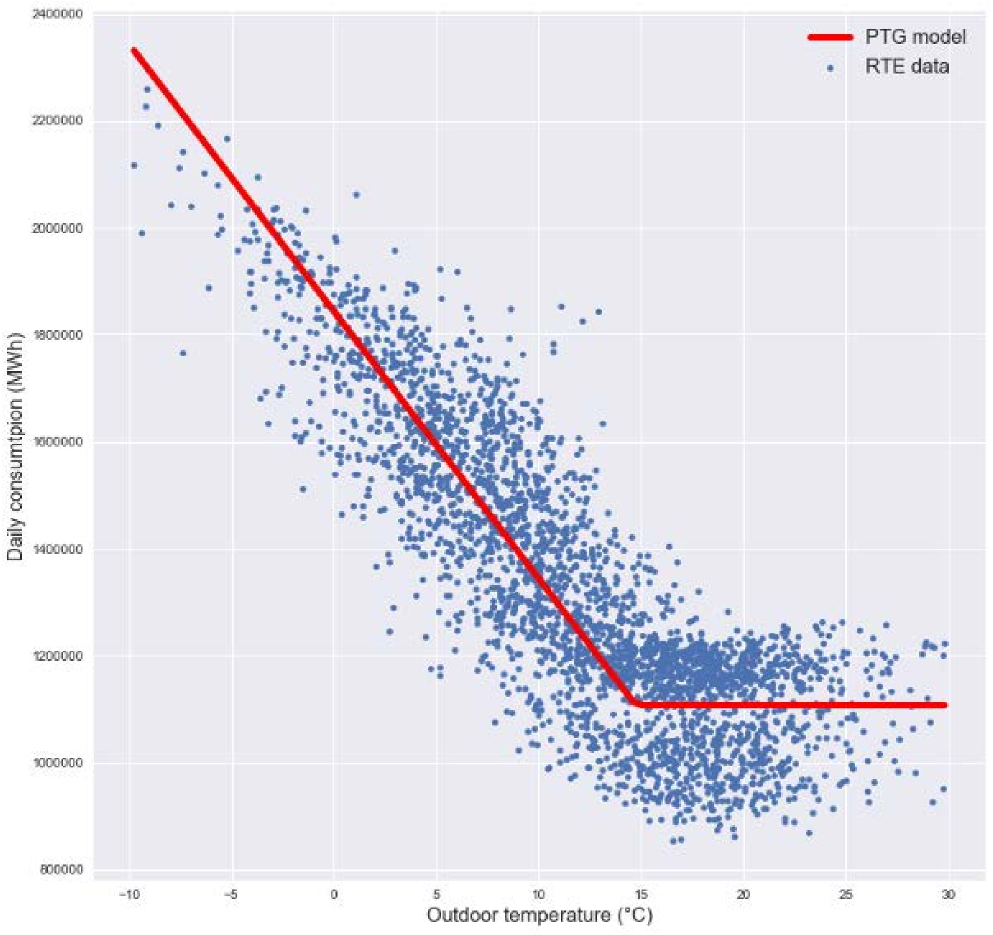


Fig. 2 - PTG forecast

In his research, he also compared a selection of different models, and optimized them to produce the best R^2 Score. He used a k-fold approach on the training set (10 folds to be exact), in this test he also came up with an R^2 Score, R^2 Score + Wind (which considered wind speed) and R^2 Score + Time (which considered both wind speed and time of day). The results are as shown in *Fig. 3*.

Models	**r²score**	**r²score(+wind)**	**r²score(+time)**
PTG	0.827		
Polynomial regressor	0.811	0.829	0.92
Random forest regressor	0.831	0.836	0.91
Decision tree regressor	0.831	0.829	0.898
K-nearest neighbour	0.823	0.831	0.931
Neural network MLP	0.829	0.84	0.904

Fig. 3 - Multiple regressor comparison

On the 10th of October 2019, a team of five students from Korea and Vietnam published a research paper titled *Improving Electricity Energy Consumption Prediction Using CNN and Bi-LSTM*. The model that they built was called The Electric Energy Consumption Prediction model utilizing the combination of CNN and Bi-LSTM Model or EECP-CBL for short.

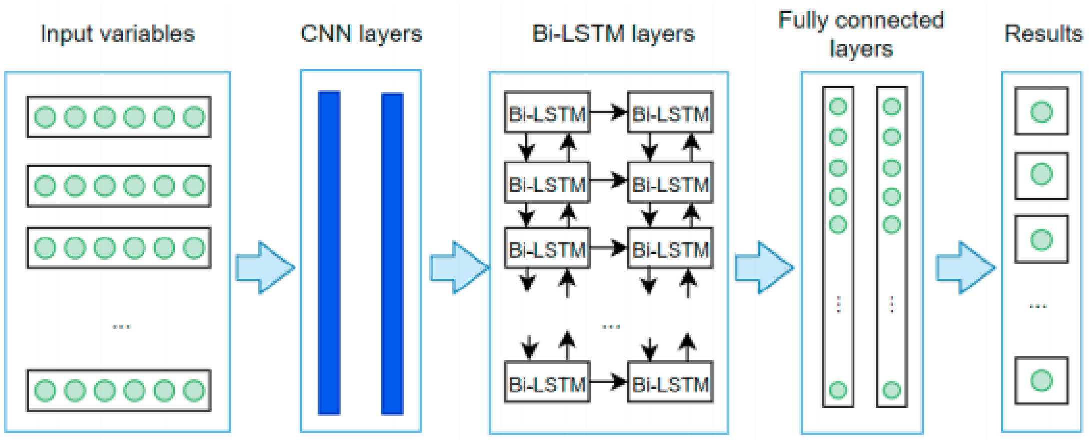


Fig. 4 - The architecture of EECP-CBL model

Fig. 4 shows the architecture of the EECP-CBL model. In this framework, two CNNs in the first module extract the important information from several variables in the Individual Household Electric Power Consumption (IHEPC) dataset. Then, the Bi-LSTM module with two Bi-LSTM layers uses the provided information as well as the trends of the time series in two directions (forward and backward states), to make the predictions. The obtained values from the Bi-LSTM module are then passed to the last module which consists of two fully connected layers for finally forecasting electric energy consumption. The results were evaluated by several performance measures for time series forecasting such as MSE, RMSE, MAE and MAPE.

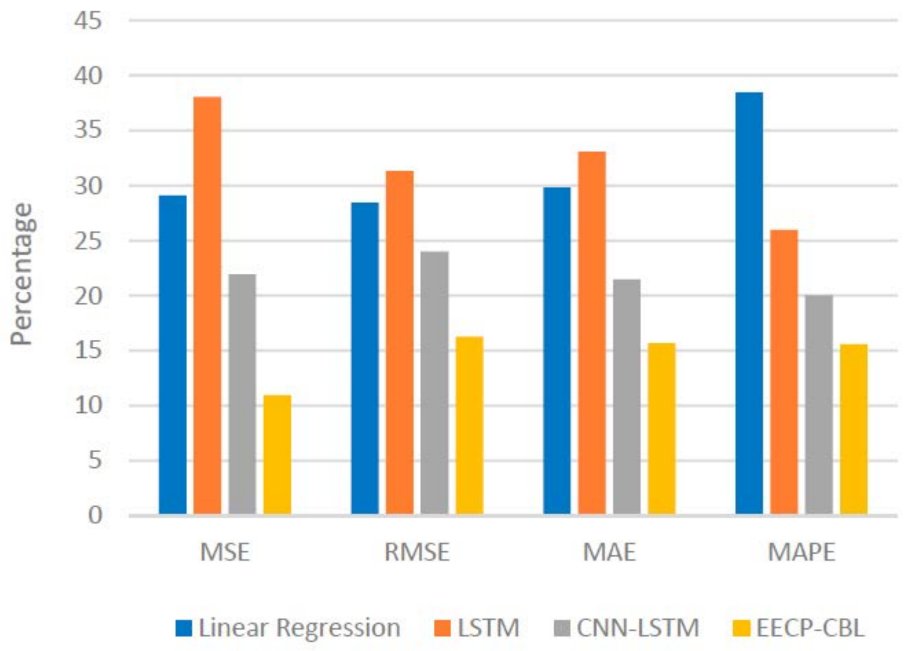


Figure 5 - The average percentages of experimental methods over four datasets

The experiments indicate that the EECP-CBL model outperforms CNN-LSTM, LSTM and Linear Regression, which are state-of-the-art models and are the common practice for electric energy consumption prediction. In terms of several performance measures such as MSE, RMSE, MAE and MAPE as well as processing time in different time frame settings, the EECP-CBL model provides the best results.

On the 23rd of November of 2001, G. Peter Zhang made a research paper on time series forecasting via the use of a hybrid Autoregressive integrated moving average (ARIMA) and neural network model. ARIMA is one of the classic econometric models and one of the most common linear models for time series forecasting, and ANN provides a promising alternative to traditional linear methods. In his paper, he implemented a hybrid methodology that combines both ARIMA and ANN models in linear and non-linear modelling. The approximation of ARIMA models to complex non-linear problems may not be adequate. On the other hand, using ANNs to model linear problems have shown to yield mixed results. Hence, why it is not wise to blindly apply ANNs to any type of data. Since time series is comprised of linear autocorrelation structure and a non-linear component, his proposed methodology for combining ARIMA and ANN is to use the ARIMA model to analyse the linear part of the problem. Afterwards, a neural network model is developed to model the residuals from the ARIMA model. Since the ARIMA model cannot capture the non-linear structure of the data, the residuals of linear model will contain information about the non-linearity. The result from the hybrid model yielded a superior forecasting accuracy than any of the models individually.

A year prior to this research paper, on the 1st of April 2000, G. Peter Zhang published a research paper on the investigation of neural networks for linear time-series forecasting.

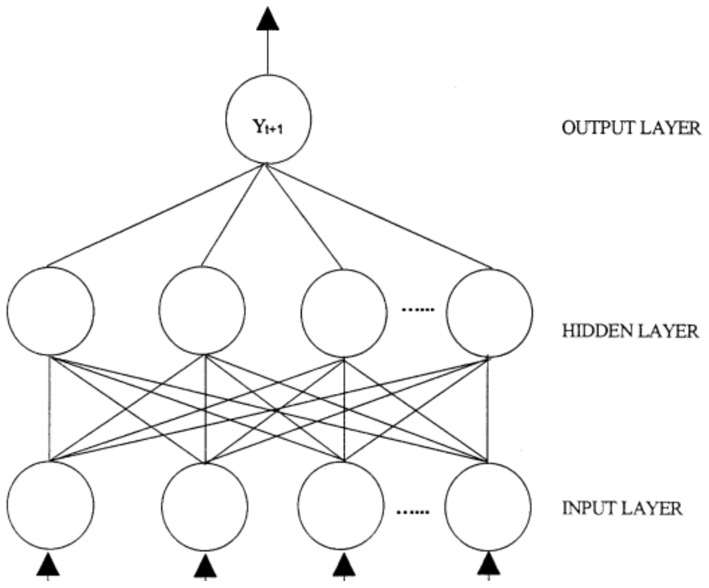


Fig. 6 - Typical fully connected feedforward neural network

The model implemented in this paper is the fully connected feedforward neural network as shown in *Fig. 6* above. For a time-series forecasting problem, past lagged observations are used as inputs. The hidden layer is composed of nodes that are connected to both the input and the output layer and is the most important part of the network. The neural network illustrated in Figure 5 is functionally a non-linear autoregressive model. This study aims to investigate the effectiveness of neural networks for time-series modelling and forecasting, and it is found that neural networks do have the competitive ability for linear time-series modelling and forecasting. Neural networks have the advantage over linear time-series models by having the ability to identify the patterns from noisy data, thus leading to a better forecast. It is also concluded from this paper that simple network models are often adequate in forecasting linear time series. This is important for practical applications since simple neural network models require much less efforts and time to build complex ones.

On the 18th of June 2020, Sarit Maitra published an article on *Vector Autoregressive for Forecasting Time Series* and the goal is to predict Brent crude oil (USA Price) from the four time-series obtained which were Brent (Europe Price), West Texas crude price and OPEC crude price. The forecast was evaluated by several performance measures, these include bias, MAE, MSE and RMSE. In this article, it was found that residuals, which are the squared difference between the predicted values and actual values, were not put to use. Based on related works, artificial neural network could be used to analyse residuals to improve accuracy. Residuals could also be used to judge the accuracy of the model, to contribute to the evaluation of its quality. The result of the forecast is as shown in *Fig. 7* below.

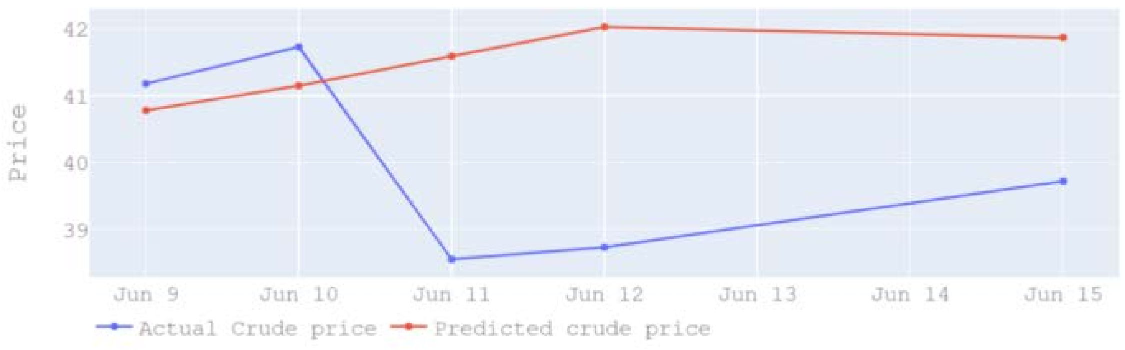


Fig. 7 - Brent crude price forecast

On the 29th of September 2020, Selcuk Disci published an article on DataGeeek that compares k nearest neighbour (KNN) regression which is a supervised machine learning method, with a more classical and stochastic process autoregressive integrated moving average (ARIMA). This article explained that it is hard to find a proper model to forecast time-series data, since models that use time-series data are often exposed to serial correlation. The results' performance measures were evaluated in the form of RMSE and MAPE. It was shown that ARIMA model is much better than the KNN model for a non-seasonal time series data.

On the 13th of April 2020, Ioannis E. Livieris, Emmanuel Pintelas and Panagiotis Pintelas published a paper on using CNN-LSTM model for gold price time-series forecasting. The proposed model consists of two main components: the first consists of convolutional and pooling layers in which complicated mathematical operations are performed to develop features of the input data, while the second component exploits the generated features by LSTM and dense layers. Convolutional neural network (CNN) is known for its ability to extract use knowledge and learn the internal representation of a time series data, while LSTM networks are effective for identifying short-term and long-term dependencies. The idea is to combine the advantages of these deep learning techniques, into a hybrid model. In conclusion although LSTM models constitute a widely accepted and efficient choice for gold price time series, their utilization along with additional convolutional layers provides a significant boost in increasing the forecasting performance. The forecast is as illustrated in *Fig. 8* below.

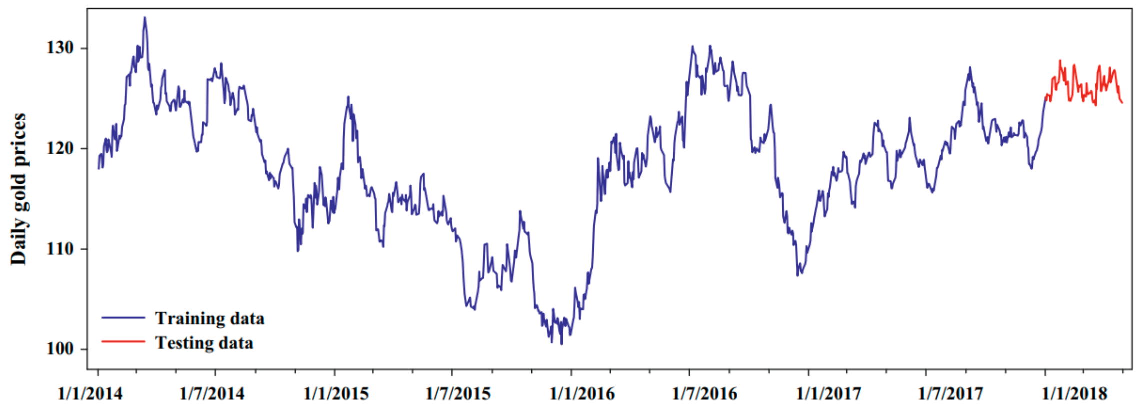


Fig. 8 - Daily gold price trend from January 2014 to April 2018

3. Description of the Work

3.1. Methodology

3.1.1. Dataset

To better follow and monitor energy consumption, the government wanted energy suppliers to install smart meters in homes in England, Wales, and Scotland. The goal is for every home to have a smart meter by 2020. The dataset used in this dissertation can be found at Kaggle. The big work of data compilation was done by Jean-Michel D. in 2019. This dataset contains the refactorized version of the data obtained from the London data store, that contains energy consumption readings for 5,567 London Households that took part in the UK Power Networks led Low Carbon London project between November of 2011 to February 2014. The data from the smart meters seems associated only to electrical consumption. The total size of this dataset when compressed in a .zip file is 2.4GB.

Readings were taken at half hourly intervals and daily intervals. Each household has been allocated a *LCLid* and a CACI Acorn group (2010). The customers in the trial were balanced samples representing the Greater London population. This dataset also contains both daily and hourly weather data provided by Dark Sky's api.

The *acorn_details.csv* file provides the detailed information on the acorn group and their profile of the people in the group. The first three columns are the attributes studied; the ACORN-X is the index of the attribute. At a national scale, the index is 100 if for one column the value is 150 it means that there are 1.5 times more people with this attribute in the ACORN group than at the national scale. The *informations_households.csv* file contains all the information on the households in the panel (their acorn group, their tariff). The *halfhourly_dataset.zip* and *daily_dataset.zip* contains the block files with the half-hourly and daily smart meter measurements, respectively. The *weatherdailydarksky.csv* and *weatherhourlydarksky.csv* file contains all the daily and the hourly weather data, respectively.

3.1.2. Preliminary Research

3.1.2.1. Support Vector Machine

Support vector machine (SVM) was originally created by Vladimir Vapnik in 1995 as a solution for two-group classification problems. It works by finding hyperplane (in multidimensional space) that separate the classes, in a two-dimensional plane this would result in a line. This is mostly used for non-linearly separable data, where there is no simple line that could separate the classes. An example of a SVM classification is as shown in *Fig.9* below.

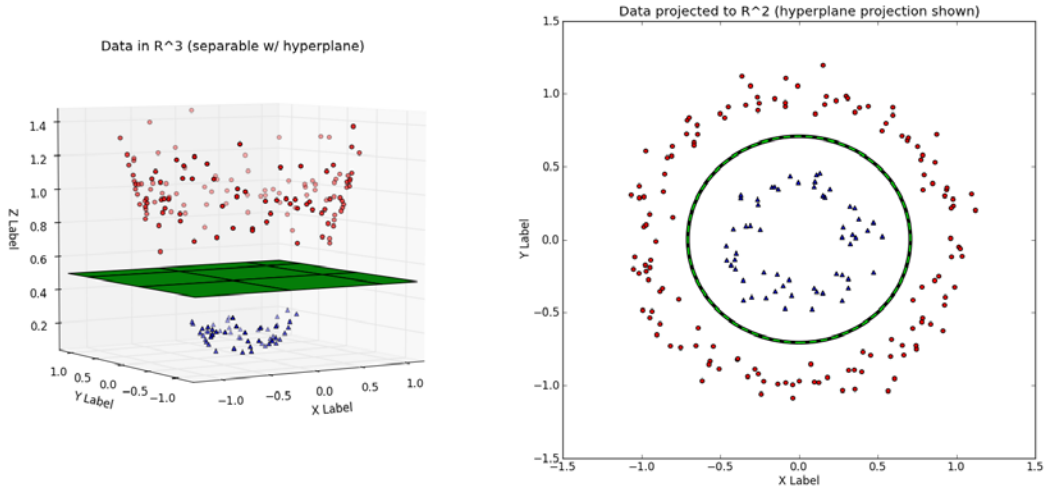


Fig. 9 - Two class SVM classification

Support vector regression (SVR) uses the same principle as SVM, but for regression problems, like time-series forecasting. The idea of SVR is to find a function that approximates mapping input domain to real numbers on the basis of a training sample. As illustrated in *Fig. 10*, consider the shaded orange line as the decision boundary, and the red line as the hyperplane. The objective of SVR is to basically consider the points within that decision boundary line. The best fit line is the hyperplane that has the maximum number of points.

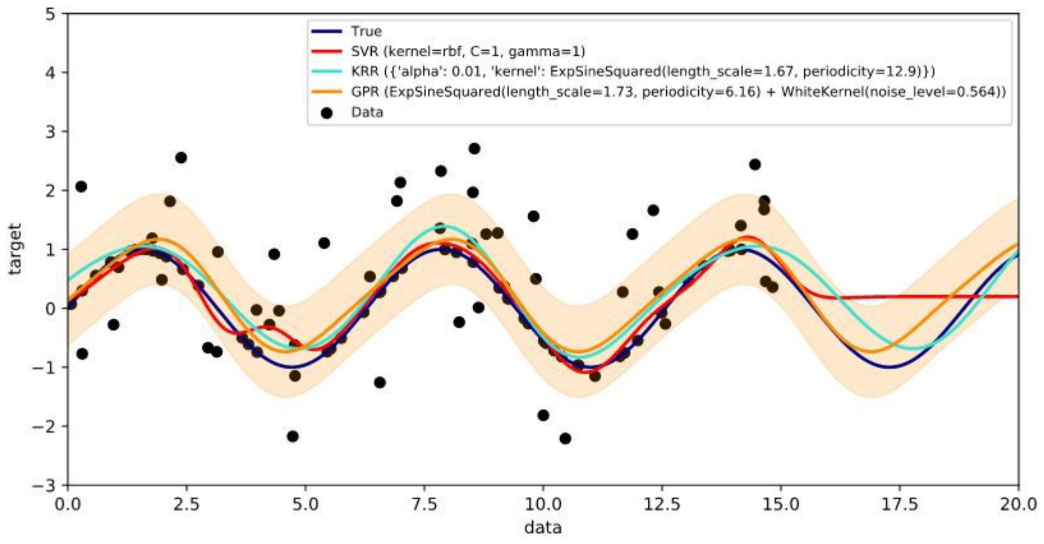


Fig. 10 - Support vector regression plot

Since SVR is derived from SVM, they take similar hyperparameters. The three main hyperparameters to SVM are the: kernel, hyperplane, and decision boundary. Kernel is a function that helps find a hyperplane in the higher dimensional space without increasing the computational cost. Usually, computational cost would increase if the dimension of the data increases, this is why SVM is very appealing as it requires less computational costs than traditional machine learning approaches. This increase in dimension is needed when no separating hyperplane can be found in a given dimension. Currently, unfortunately Scikit's SVR package is not yet optimally supported for multivariate time-series forecasting. It is still mainly designed for univariate time series forecasting, where the time series refers to one that consists of single observations recorded sequentially over equal time increments. On the other hand, multivariate time series model is an extension of the univariate case and involves two or more input variables. There are three main kernel functions for SVR that I have chosen to implement in this project, these are: linear, polynomial, and radial basis function (RBF) kernels. These kernels were chosen due to them being a general-purpose function used in implementations where no prior knowledge about the data is available, and to find out how suitable are they for time-series forecasting.

$$Linear\ Kernel \rightarrow K_s(x, x_i) = x.x_i$$

Linear kernel as shown above, is a function used when the data is linearly separable (able to be separated by a single line) and is one of the most common kernels to be used when there are a large number of features present in a particular data set.

$$Polynomial\ Kernel \rightarrow\ K(x, x') = (1 + x * x')^k$$

Polynomial kernel function as shown above, represents the similarity of training samples in a feature space over polynomials of the original variables. This allows it to learn non-linear models, like time series forecasting.

$$RBF\ Kernel \rightarrow K(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{0.01}\right)$$

RBF kernel as shown above, is a kernel function where the value depends on the distance from the origin or from a certain point. RBF kernel is popular due to its similarity with K-Nearest Neighbour algorithm. It has the advantages of K-NN but overcomes the complexity problem as RBF stores the support vectors during training and not the entire dataset.

3.1.2.2. Vector Autoregression

Vector autoregression (VAR) is a multivariate forecasting algorithm that is used when multiple time-series influence each other. It is considered an autoregressive model because each time series (variable) is modelled as a function of the past values, meaning that the predictors are the lags (time delayed value) of the series. A *p*th-order VAR model would be written as such:

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t$$

Where p is the number of lags, t is the period of time, y_t is the vector where a set of k variables are stored in. As illustrated in *Fig. 11*, the orange line is the forecast with the dashed line being the plotted confidence intervals.

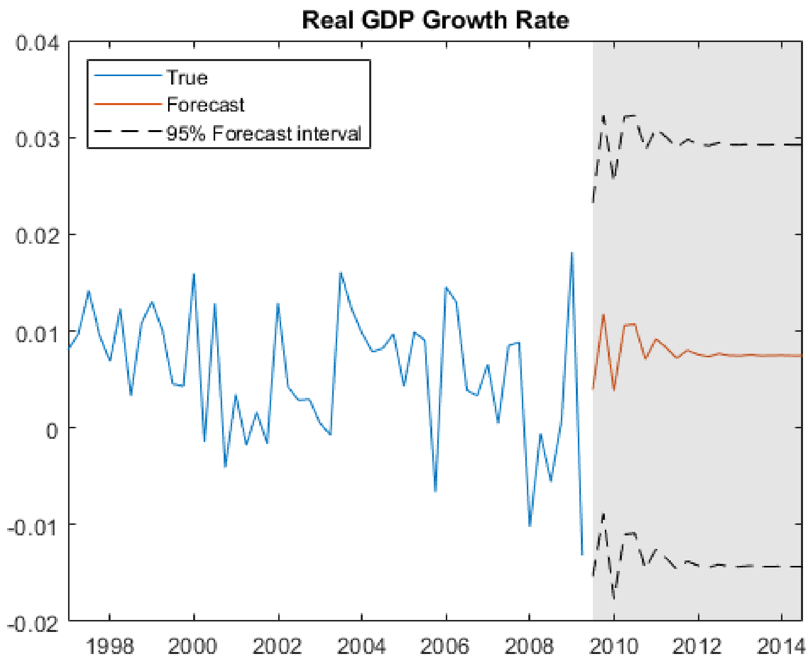


Fig. 11 - Vector autoregression plot

3.1.3. Evaluation Metrics

$$Bias = \sum_{t=0}^{T} (Y_t - F_t)/T$$

A forecast bias occurs when there is a tendency for a forecast to be consistently higher or lower than the actual value. Forecast bias is distinct from forecast error in that a forecast can have any level of error but still be completely unbiased. A positive forecast bias indicates over forecasting and a negative forecast bias indicates under forecasting.

$$MAPE = \frac{\sum_{i=0}^{n} |y_i - \hat{y}_i|}{n} x100\%$$

The mean absolute percentage error (MAPE) is a measure of how accurate a forecast system is. It measures this accuracy as a percentage and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values.

$$RMSE = \sqrt{mean(e_t^2)}$$

Root Mean Squared Error (RMSE) shows scale-dependent errors. It is the standard deviation of prediction errors. Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out their residuals are. In essence, it shows you how concentrated the data is around the line of best fit.

$$MAE = \frac{\sum_{i=0}^{n} |y_i - \hat{y}_i|}{n}$$

Mean absolute error is the average of the absolute values of the deviation. This type of error measurement is useful when measuring prediction errors in the same unit as the original series. It shows how big of an error you can expect from the forecast in average.

3.2. Design

3.2.1. Dataset

The dataset originally comes with more than 20 variables. There were a few criteria applied for feature selection (choosing the final number of variables in the dataset). These are, the presence of redundant variables, variables with *NaN* entries, variables that clearly has no effect on energy consumption and most importantly variables that have no correlation energy consumption. The test performed on the variables is the Granger-Causality test. This is important for multivariate time-series forecasting, where correlation between the variables is considered for. However, before proceeding with the Granger-Causality test a check for stationarity is done since Granger-Causality tests require the data to be stationary. A stationarity data is data that have mean, variance and autocorrelation structure that do not change over time. For the purpose of this project, data with a flat looking series, without trend, constant variance over time, a constant autocorrelation structure over time and no periodic fluctuations are ideal. In other words, we do not want seasonality. Stationarity is important for both SVR and VAR models. To check for stationarity a visual check and an AD-Fuller test was performed. *Fig. 12, Fig. 13, Fig. 14,* and *Fig. 15* below are the two samples of data plotted.

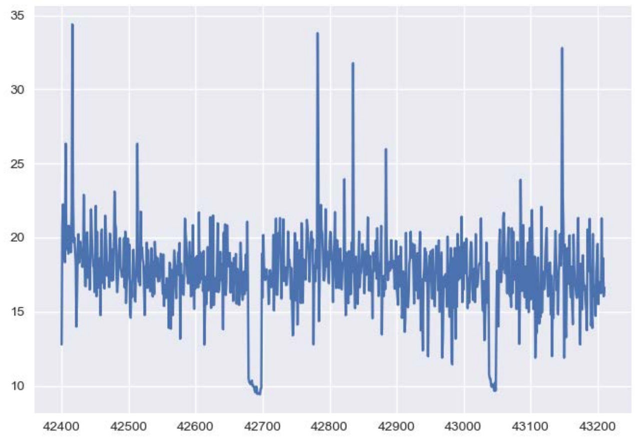


Fig. 14 - Acorn G data sample

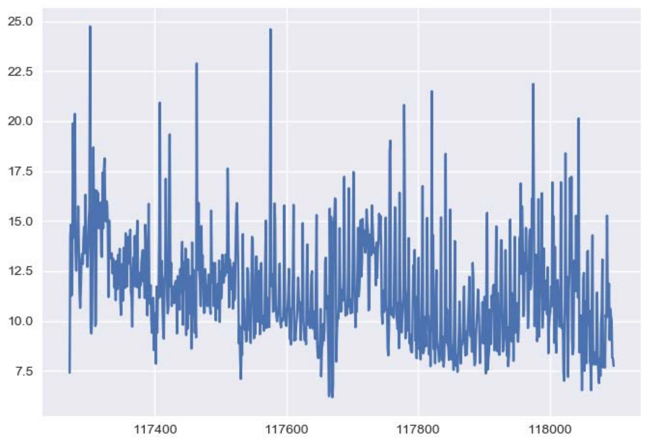


Fig. 13 - Acorn H data sample

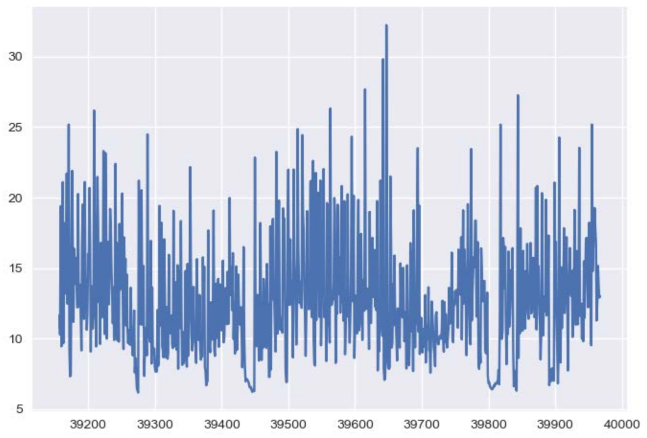


Fig. 12 - Acorn K data sample

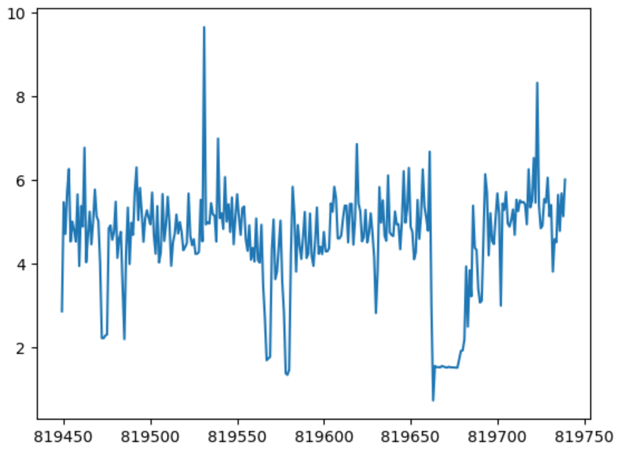
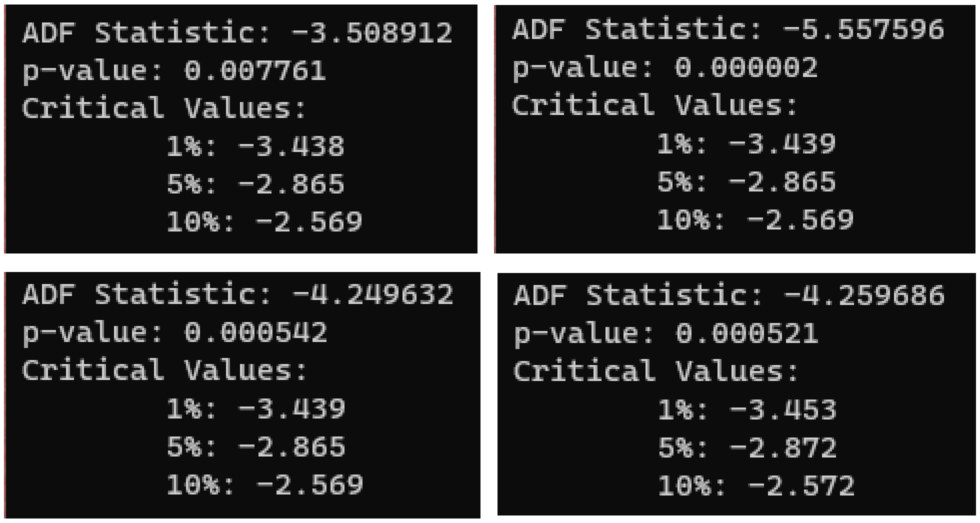


Fig. 15 - Acorn I data sample

From this visual check, it can be concluded that there are little to no seasonality present in the data. Little to no trends are visible. Households (*LCLid*) with obvious trends and seasonality are dropped. However, this type of visual check can be considered subjective. That is why an AD-Fuller test was performed, for a more consistent result.



```
ADF Statistic: -3.508912
p-value: 0.007761
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.569
```

```
ADF Statistic: -5.557596
p-value: 0.000002
Critical Values:
        1%: -3.439
        5%: -2.865
        10%: -2.569
```

```
ADF Statistic: -4.249632
p-value: 0.000542
Critical Values:
        1%: -3.439
        5%: -2.865
        10%: -2.569
```

```
ADF Statistic: -4.259686
p-value: 0.000521
Critical Values:
        1%: -3.453
        5%: -2.872
        10%: -2.572
```

Fig. 16 - ADF results for sample of acorn G, H, K, I

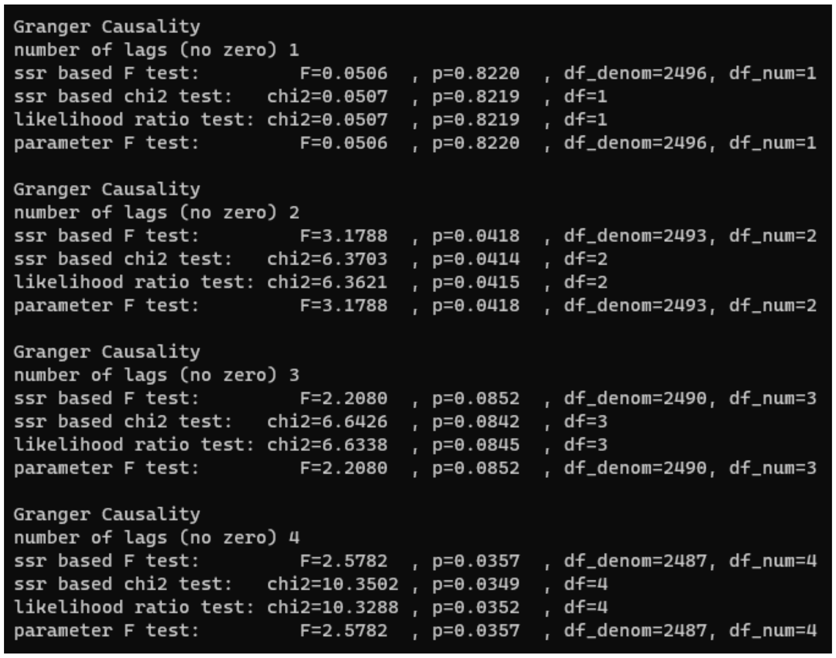
Fig. 16 shown above are the results for the AD-Fuller check. The lower the ADF statistics value, the more stationary it is. For the purpose of this project, we want our ADF statistic value to be less than our critical values. The AD-Fuller test was performed on 15 randomly selected household from every acorn group. For the purpose of the paper, we will be analyzing just four of the different acorn group samples (G, H, K, I), however the process performed on these four samples were identical throughout all the different type of acorn groups (A-P & U).

3.2.2. Support Vector Regression

For the purpose of this project, three types of SVR kernel functions were implemented for comparison. These are the linear, polynomial, and radial basis function. Since this will be in the form of a univariate time-series forecast, we only include the energy consumption data and index (time), the rest of the features are dropped. Before these models are fitted, a feature scaling function was applied to the features. This is to ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features. Afterwards, the results from all three models were predicted and plotted in the form of a time-series graph.

3.2.3. Vector Autoregression

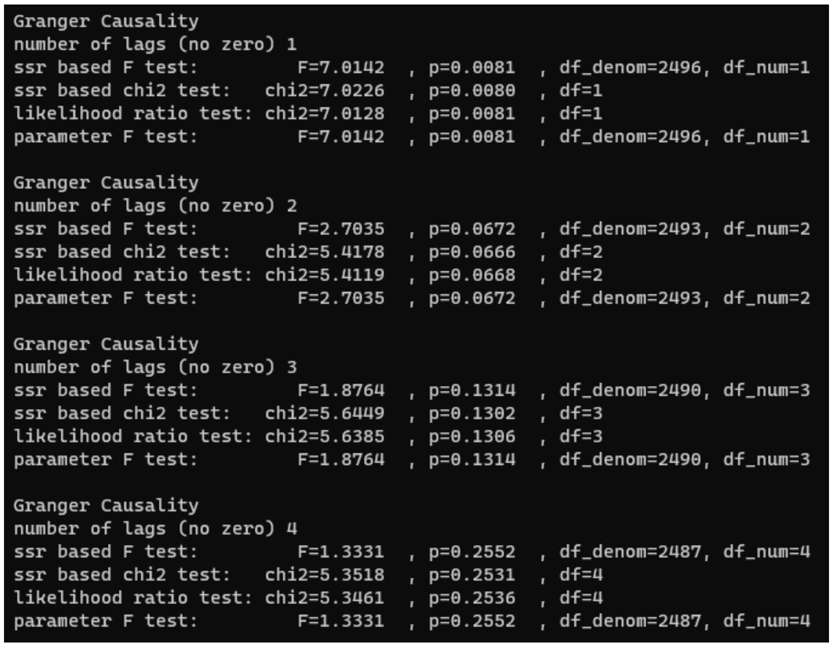
Before fitting the model, feature selection is needed as vector autoregression models rely heavily in autocorrelation. After performing AD-Fuller test on all 18 samples (one for each acorn group), a Granger-Causality test was performed on all of the feautres for the purpose of feature selection. These variables were performed Granger-Causality test against energy consumption. The two output values that are taken into consideration is the p-value and the number of lags. If the p-value is small, we can reject the null hypothesis. The threshold of the p-value was set at 0.075, meaning that if the value of p is less than 0.075, the null hypothesis is rejected. In this case, the null hypothesis states that a given variable Granger-cause energy consumption at a certain lag. The maximum number of lags performed on each of these variables was four and the results of the Granger-Causality tests are as shown in *Fig. 17 - Fig. 21* below.



```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.0506  , p=0.8220  , df_denom=2496, df_num=1
ssr based chi2 test:   chi2=0.0507  , p=0.8219  , df=1
likelihood ratio test: chi2=0.0507  , p=0.8219  , df=1
parameter F test:         F=0.0506  , p=0.8220  , df_denom=2496, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=3.1788  , p=0.0418  , df_denom=2493, df_num=2
ssr based chi2 test:   chi2=6.3703  , p=0.0414  , df=2
likelihood ratio test: chi2=6.3621  , p=0.0415  , df=2
parameter F test:         F=3.1788  , p=0.0418  , df_denom=2493, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=2.2080  , p=0.0852  , df_denom=2490, df_num=3
ssr based chi2 test:   chi2=6.6426  , p=0.0842  , df=3
likelihood ratio test: chi2=6.6338  , p=0.0845  , df=3
parameter F test:         F=2.2080  , p=0.0852  , df_denom=2490, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=2.5782  , p=0.0357  , df_denom=2487, df_num=4
ssr based chi2 test:   chi2=10.3502 , p=0.0349  , df=4
likelihood ratio test: chi2=10.3288 , p=0.0352  , df=4
parameter F test:         F=2.5782  , p=0.0357  , df_denom=2487, df_num=4
```

Fig. 17 - Humidity Granger-Causality test

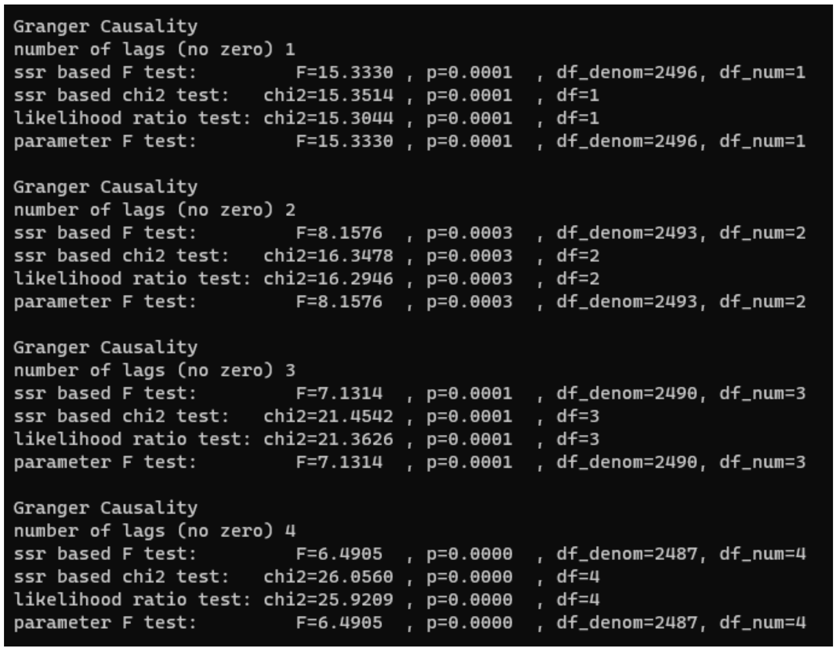
Fig. 17 above shows the Granger-Causality test result of humidity against energy consumption. It is seen that at lag 2, the p-value is within the threshold of 0.075. This means that we can reject the null hypothesis, and say that at lag 2, humidity Granger-cause energy consumption.



```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=7.0142  , p=0.0081  , df_denom=2496, df_num=1
ssr based chi2 test:   chi2=7.0226  , p=0.0080  , df=1
likelihood ratio test: chi2=7.0128  , p=0.0081  , df=1
parameter F test:         F=7.0142  , p=0.0081  , df_denom=2496, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=2.7035  , p=0.0672  , df_denom=2493, df_num=2
ssr based chi2 test:   chi2=5.4178  , p=0.0666  , df=2
likelihood ratio test: chi2=5.4119  , p=0.0668  , df=2
parameter F test:         F=2.7035  , p=0.0672  , df_denom=2493, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=1.8764  , p=0.1314  , df_denom=2490, df_num=3
ssr based chi2 test:   chi2=5.6449  , p=0.1302  , df=3
likelihood ratio test: chi2=5.6385  , p=0.1306  , df=3
parameter F test:         F=1.8764  , p=0.1314  , df_denom=2490, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=1.3331  , p=0.2552  , df_denom=2487, df_num=4
ssr based chi2 test:   chi2=5.3518  , p=0.2531  , df=4
likelihood ratio test: chi2=5.3461  , p=0.2536  , df=4
parameter F test:         F=1.3331  , p=0.2552  , df_denom=2487, df_num=4
```

Fig. 18 - UV Index Granger-Causality test

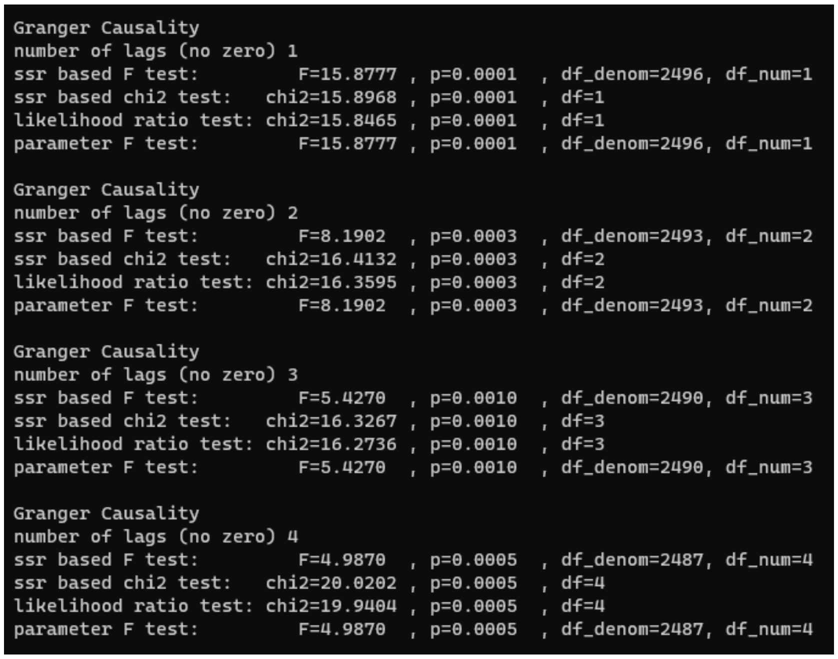
Fig. 18 above shows the Granger-Causality test result of UV Index against energy consumption. It is seen that at lag 2, the p-value is within the threshold of 0.075. This means that we can reject the null hypothesis, and say that at lag 2, UV Index Granger-cause energy consumption.



```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=15.3330 , p=0.0001  , df_denom=2496, df_num=1
ssr based chi2 test:   chi2=15.3514 , p=0.0001  , df=1
likelihood ratio test: chi2=15.3044 , p=0.0001  , df=1
parameter F test:         F=15.3330 , p=0.0001  , df_denom=2496, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=8.1576  , p=0.0003  , df_denom=2493, df_num=2
ssr based chi2 test:   chi2=16.3478 , p=0.0003  , df=2
likelihood ratio test: chi2=16.2946 , p=0.0003  , df=2
parameter F test:         F=8.1576  , p=0.0003  , df_denom=2493, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=7.1314  , p=0.0001  , df_denom=2490, df_num=3
ssr based chi2 test:   chi2=21.4542 , p=0.0001  , df=3
likelihood ratio test: chi2=21.3626 , p=0.0001  , df=3
parameter F test:         F=7.1314  , p=0.0001  , df_denom=2490, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=6.4905  , p=0.0000  , df_denom=2487, df_num=4
ssr based chi2 test:   chi2=26.0560 , p=0.0000  , df=4
likelihood ratio test: chi2=25.9209 , p=0.0000  , df=4
parameter F test:         F=6.4905  , p=0.0000  , df_denom=2487, df_num=4
```

Fig. 19 - Minimum temperature Granger-Causality test

Fig. 19 above shows the Granger-Causality test result of the minimum temperature against energy consumption. It is seen that at lag 1 to 4, the p-value is well within the threshold of 0.075. This means that we can reject the null hypothesis, and say that, the minimum tempeature Granger-cause energy consumption.



```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=15.8777 , p=0.0001  , df_denom=2496, df_num=1
ssr based chi2 test:   chi2=15.8968 , p=0.0001  , df=1
likelihood ratio test: chi2=15.8465 , p=0.0001  , df=1
parameter F test:         F=15.8777 , p=0.0001  , df_denom=2496, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=8.1902  , p=0.0003  , df_denom=2493, df_num=2
ssr based chi2 test:   chi2=16.4132 , p=0.0003  , df=2
likelihood ratio test: chi2=16.3595 , p=0.0003  , df=2
parameter F test:         F=8.1902  , p=0.0003  , df_denom=2493, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=5.4270  , p=0.0010  , df_denom=2490, df_num=3
ssr based chi2 test:   chi2=16.3267 , p=0.0010  , df=3
likelihood ratio test: chi2=16.2736 , p=0.0010  , df=3
parameter F test:         F=5.4270  , p=0.0010  , df_denom=2490, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=4.9870  , p=0.0005  , df_denom=2487, df_num=4
ssr based chi2 test:   chi2=20.0202 , p=0.0005  , df=4
likelihood ratio test: chi2=19.9404 , p=0.0005  , df=4
parameter F test:         F=4.9870  , p=0.0005  , df_denom=2487, df_num=4
```

Fig. 20 - Maximum temperature Granger-Causality test

Fig. 20 above shows the Granger-Causality test result of the maximum temperature against energy consumption. It is seen that at lag 1 to 4, the p-value is well within the threshold of 0.075. This means that we can reject the null hypothesis, and say that, the maximum tempeature Granger-cause energy consumption.

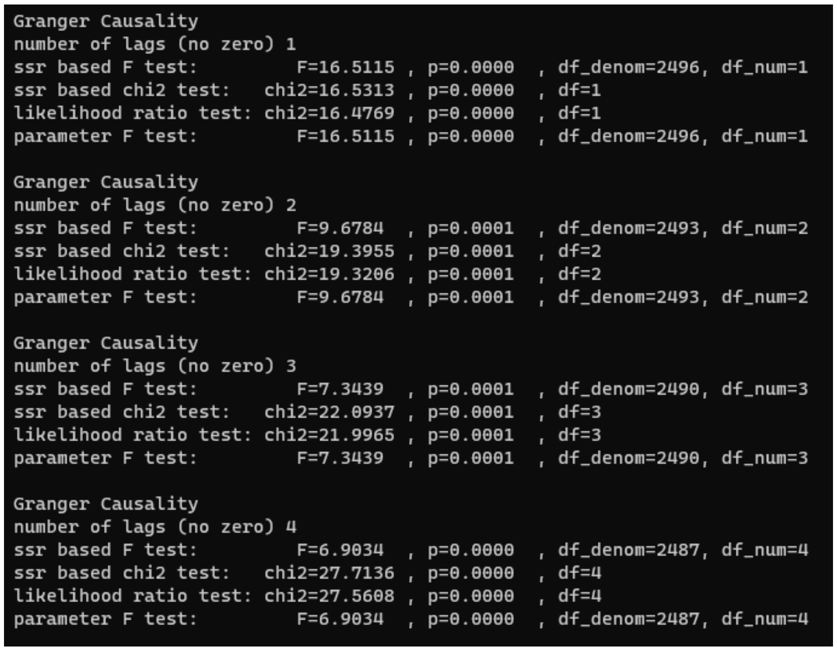

```
Granger Causality
number of lags (no zero) 1
ssr based F test:         F=16.5115 , p=0.0000  , df_denom=2496, df_num=1
ssr based chi2 test:   chi2=16.5313 , p=0.0000  , df=1
likelihood ratio test: chi2=16.4769 , p=0.0000  , df=1
parameter F test:         F=16.5115 , p=0.0000  , df_denom=2496, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=9.6784  , p=0.0001  , df_denom=2493, df_num=2
ssr based chi2 test:   chi2=19.3955 , p=0.0001  , df=2
likelihood ratio test: chi2=19.3206 , p=0.0001  , df=2
parameter F test:         F=9.6784  , p=0.0001  , df_denom=2493, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=7.3439  , p=0.0001  , df_denom=2490, df_num=3
ssr based chi2 test:   chi2=22.0937 , p=0.0001  , df=3
likelihood ratio test: chi2=21.9965 , p=0.0001  , df=3
parameter F test:         F=7.3439  , p=0.0001  , df_denom=2490, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=6.9034  , p=0.0000  , df_denom=2487, df_num=4
ssr based chi2 test:   chi2=27.7136 , p=0.0000  , df=4
likelihood ratio test: chi2=27.5608 , p=0.0000  , df=4
parameter F test:         F=6.9034  , p=0.0000  , df_denom=2487, df_num=4
```

Fig. 21 - Mean temperature Granger-Causality test

Fig. 21 above shows the Granger-Causality test result of the mean temperature against energy consumption. It is seen that at lag 1 to 4, the p-value is well within the threshold of 0.075. This means that we can reject the null hypothesis, and say that, the mean tempeature Granger-cause energy consumption.

Since the mean temperature is a product of the maximum and minimum temperature, for the final shape of the dataset, it is decided that they would be dropped. The final shape of the dataset is as shown on *Table 1* below.

Table 1 - Dataset Description

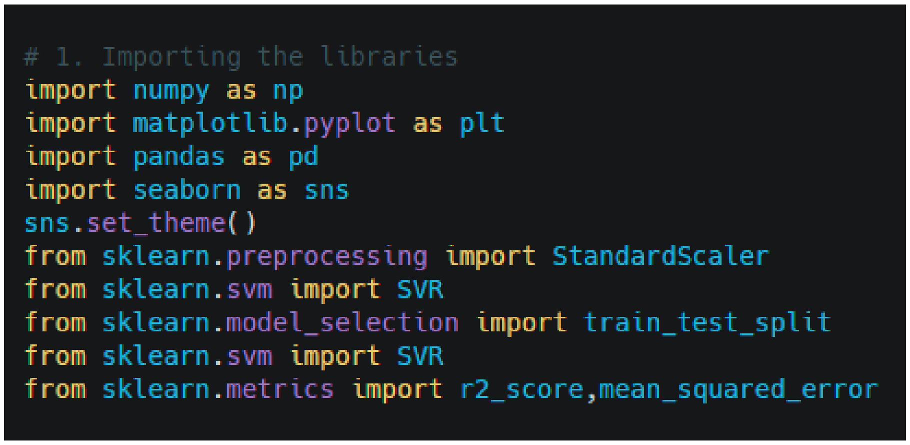
Variable	Description
energy_sum	The total energy consumption of a household in a given day. The unit is Kilowatt Hour (KWh).
LCLid	The ID given for every household. The format is 'MAC******'.
Date	The date for a given day. The format is YYYY-MM-DD.
temperatureMean	The average temperature in London at a given day. The unit is Celsius (C°).

humidity	The humidity in London at a given day. The unit is in percentage (%).
uvIndex	The uvIndex in London at a given day.
Acorn	The acorn given to an *LCLid* (A-P & U).

4. Implementation

4.1. Support Vector Regression

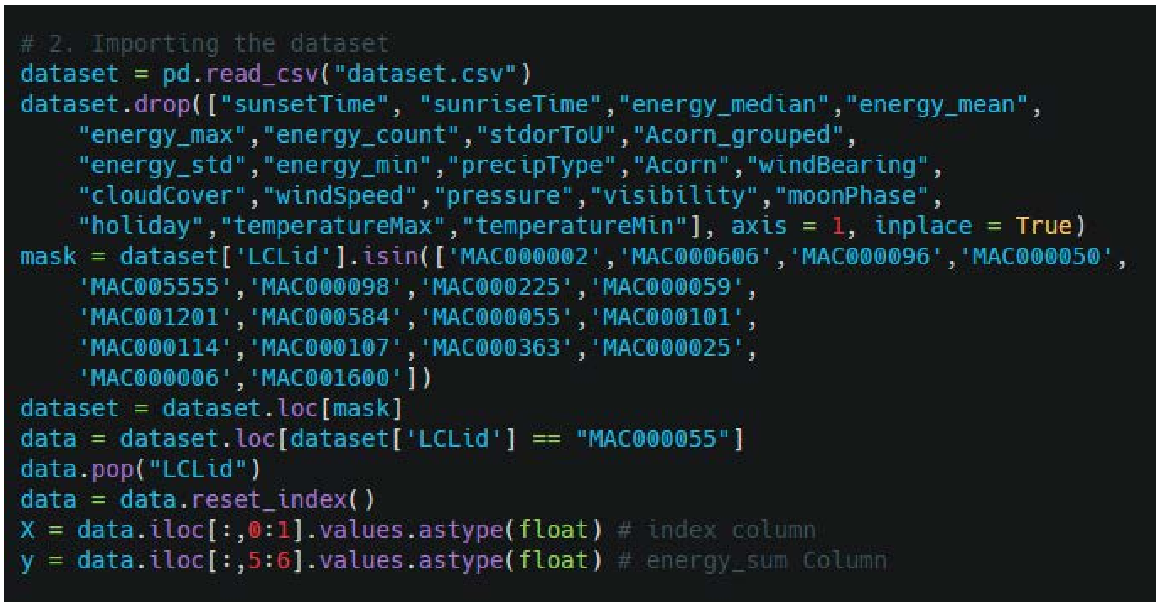
The language used for the development of this project is Python and was run on a Conda virtual environment, due to its simple syntax, and it is the language I am most familiar with. Python also has an active community with a vast selection of libraries and resources. The dependencies that are required for the SVR implementation are as shown in *Fig. 22* below.



```python
# 1. Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
sns.set_theme()
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import r2_score,mean_squared_error
```

Fig. 22 - Importing dependencies (SVR)

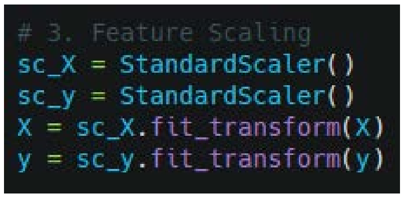
NumPy, which stands for Numerical Python is a Python library used for working with arrays. It offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. In this project NumPy was used for array manipulation. Matplotlib is the package that will be used for data visualization since it is able to not only plot basic data types but also NumPy arrays and Pandas Dataframe. Similar to NumPy, Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Dataframe. Scikit-learn or Sklearn is a machine learning and statistical modelling library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours.



```
# 2. Importing the dataset
dataset = pd.read_csv("dataset.csv")
dataset.drop(["sunsetTime", "sunriseTime","energy_median","energy_mean",
    "energy_max","energy_count","stdorToU","Acorn_grouped",
    "energy_std","energy_min","precipType","Acorn","windBearing",
    "cloudCover","windSpeed","pressure","visibility","moonPhase",
    "holiday","temperatureMax","temperatureMin"], axis = 1, inplace = True)
mask = dataset['LCLid'].isin(['MAC000002','MAC000606','MAC000096','MAC000050',
    'MAC005555','MAC000098','MAC000225','MAC000059',
    'MAC001201','MAC000584','MAC000055','MAC000101',
    'MAC000114','MAC000107','MAC000363','MAC000025',
    'MAC000006','MAC001600'])
dataset = dataset.loc[mask]
data = dataset.loc[dataset['LCLid'] == "MAC000055"]
data.pop("LCLid")
data = data.reset_index()
X = data.iloc[:,0:1].values.astype(float) # index column
y = data.iloc[:,5:6].values.astype(float) # energy_sum Column
```

Fig. 23 - Importing the dataset (SVR)

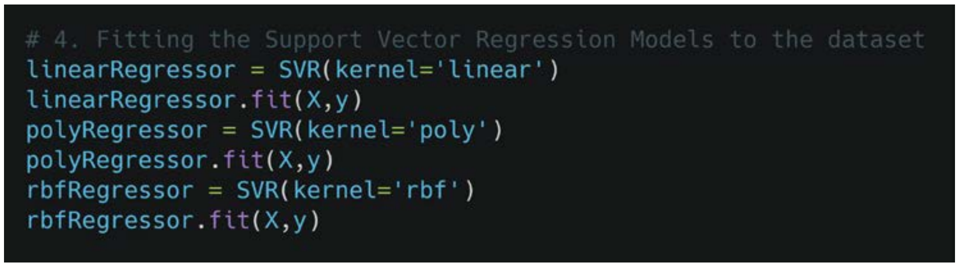
After importing all the necessary dependencies, the dataset is imported. Unused features are dropped and the randomly selected households (*LCLid*) from each acorn group is isolated. After the *LCLid* is selected, the index and energy_sum feature is assigned to variable X, and y respectively as shown in *Fig. 23* above.



```
# 3. Feature Scaling
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

Fig. 24 - Feature scaling (SVR)

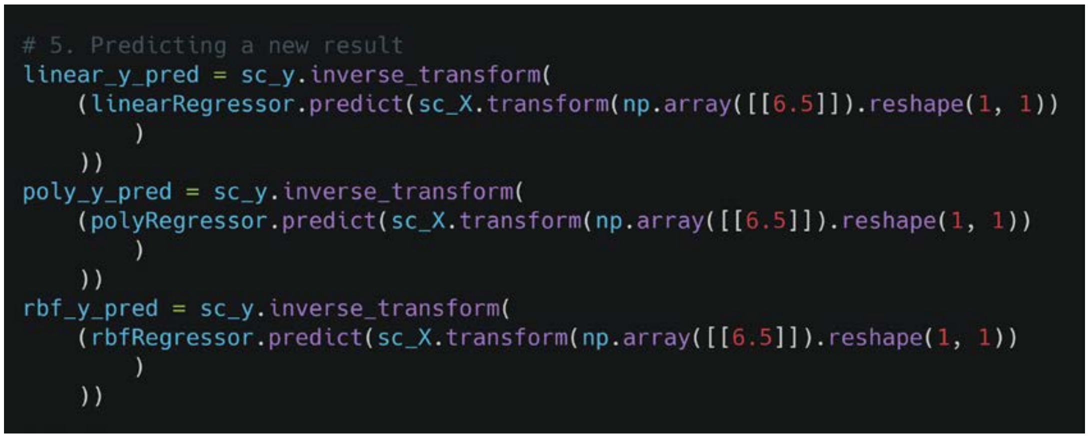
Before fitting the model, feature scaling is applied to the variables as shown in *Fig. 24* above. The *.fit_transform()* function is used on the training data to scale and learn the scaling parameters of that data. These learned parameters are then used to scale the test data.



```
# 4. Fitting the Support Vector Regression Models to the dataset
linearRegressor = SVR(kernel='linear')
linearRegressor.fit(X,y)
polyRegressor = SVR(kernel='poly')
polyRegressor.fit(X,y)
rbfRegressor = SVR(kernel='rbf')
rbfRegressor.fit(X,y)
```

Fig. 25 - Model fitting (SVR)

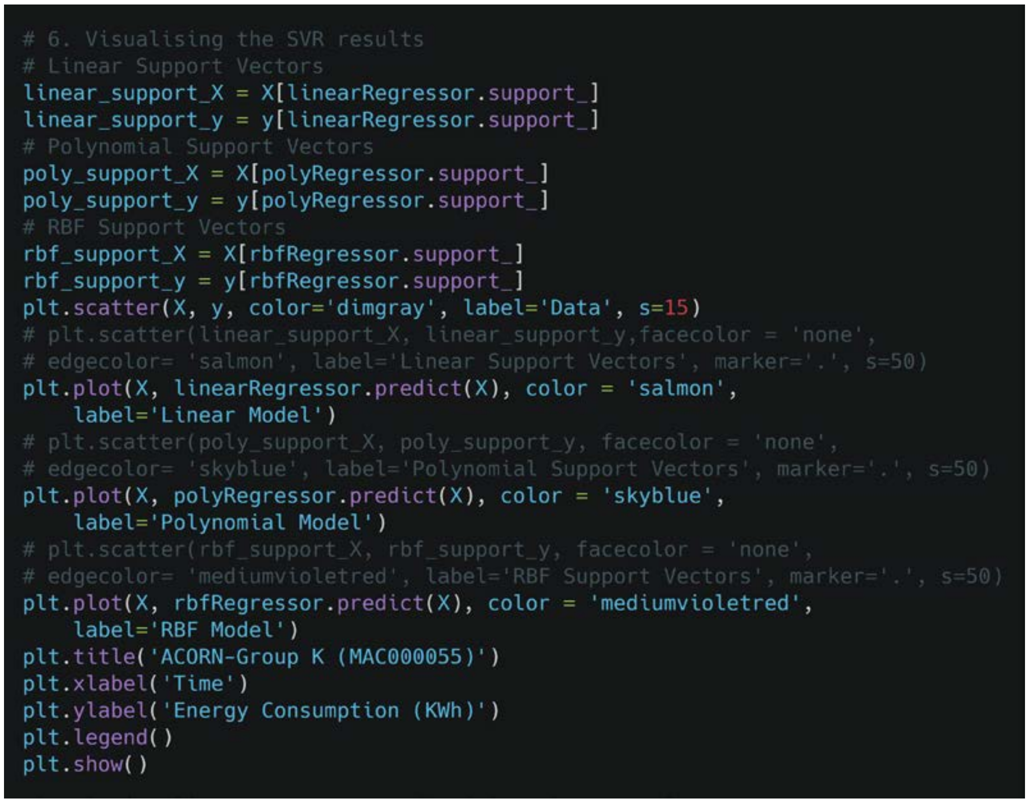
Three separate regressors for each kernel functions are then created, by calling the *.SVR()* function and specifying the kernel function. Afterwards, the respective SVR models can be fitted as shown in *Fig. 25* above.



```
# 5. Predicting a new result
linear_y_pred = sc_y.inverse_transform(
    (linearRegressor.predict(sc_X.transform(np.array([[6.5]]).reshape(1, 1))
        )
    ))
poly_y_pred = sc_y.inverse_transform(
    (polyRegressor.predict(sc_X.transform(np.array([[6.5]]).reshape(1, 1))
        )
    ))
rbf_y_pred = sc_y.inverse_transform(
    (rbfRegressor.predict(sc_X.transform(np.array([[6.5]]).reshape(1, 1))
        )
    ))
```

Fig. 26 - Predicting the result (SVR)

The results for each regressor are then predicted by calling the *.predict()* function as shown in *Fig. 26* above. The *.inverse_transform()* function is used to get the original unscaled data back.



```
# 6. Visualising the SVR results
# Linear Support Vectors
linear_support_X = X[linearRegressor.support_]
linear_support_y = y[linearRegressor.support_]
# Polynomial Support Vectors
poly_support_X = X[polyRegressor.support_]
poly_support_y = y[polyRegressor.support_]
# RBF Support Vectors
rbf_support_X = X[rbfRegressor.support_]
rbf_support_y = y[rbfRegressor.support_]
plt.scatter(X, y, color='dimgray', label='Data', s=15)
# plt.scatter(linear_support_X, linear_support_y,facecolor = 'none',
# edgecolor= 'salmon', label='Linear Support Vectors', marker='.', s=50)
plt.plot(X, linearRegressor.predict(X), color = 'salmon',
    label='Linear Model')
# plt.scatter(poly_support_X, poly_support_y, facecolor = 'none',
# edgecolor= 'skyblue', label='Polynomial Support Vectors', marker='.', s=50)
plt.plot(X, polyRegressor.predict(X), color = 'skyblue',
    label='Polynomial Model')
# plt.scatter(rbf_support_X, rbf_support_y, facecolor = 'none',
# edgecolor= 'mediumvioletred', label='RBF Support Vectors', marker='.', s=50)
plt.plot(X, rbfRegressor.predict(X), color = 'mediumvioletred',
    label='RBF Model')
plt.title('ACORN-Group K (MAC000055)')
plt.xlabel('Time')
plt.ylabel('Energy Consumption (KWh)')
plt.legend()
plt.show()
```

Fig. 27 - Results visualization (SVR)

The *.support()* function returns the data points for all the support vectors for a given model. This is useful when plotting the support vector is necessary.

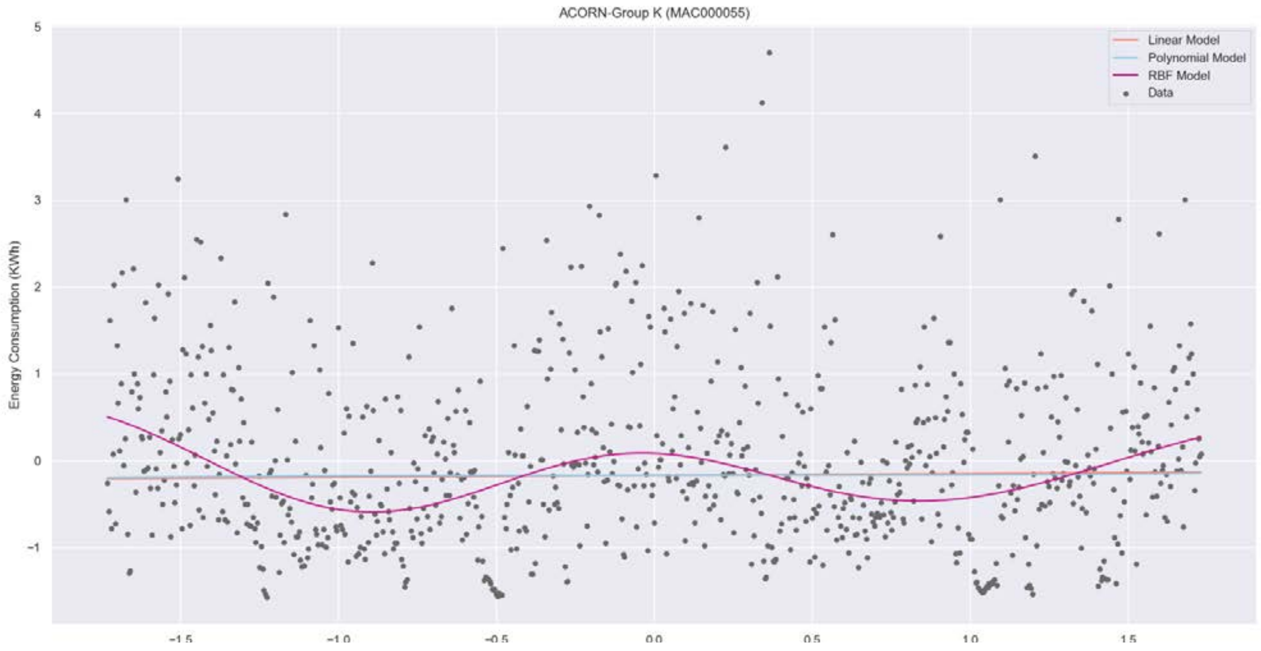
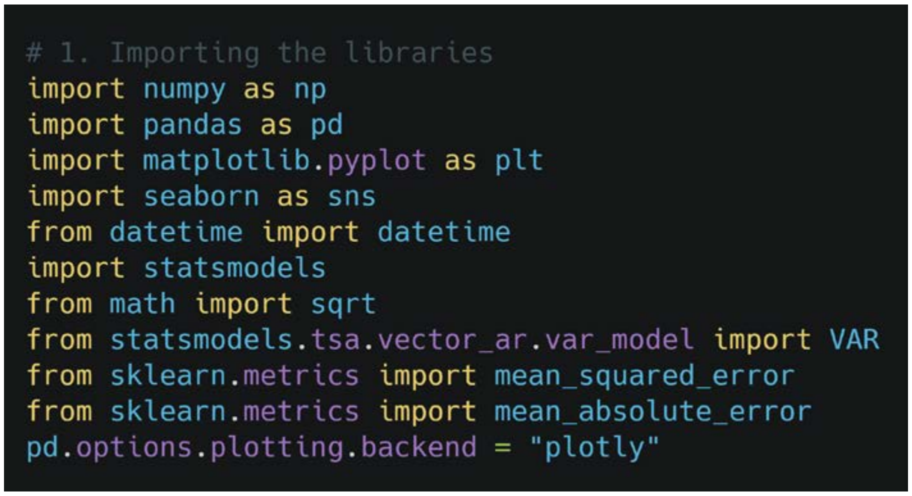


Fig. 28 – Plotting (SVR)

The time-series plot is as shown in *Fig. 28* above. The orange-coloured plot represents the linear model, the light, blue-coloured plot represents the polynomial model, the pink-coloured plot represents the RBF model, and the black scatter plots are the actual. We can see that the RBF model creates the best fitting hyperplane, compared to the linear and polynomial model.

4.2. Vector Autoregression



```python
# 1. Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import statsmodels
from math import sqrt
from statsmodels.tsa.vector_ar.var_model import VAR
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
pd.options.plotting.backend = "plotly"
```

Fig. 29 - Importing dependencies (VAR)

The dependencies that are required for the VAR implementation are as shown in *Fig. 29* below. VAR requires a different package from SVR, instead of using Scikit-learn, Statsmodels is used. Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, statistical tests, and statistical data exploration. Scikit-learn is still used for performance measures evaluation.

After importing all the necessary dependencies, the dataset is imported. Unused features are dropped and the randomly selected households (*LCLid*) from each acorn group is isolated. The selected features are *humidity, mean temperature, energy consumption, UV index, date,* and *LCLid*. After the variables of a given *LCLid* is selected, the *LCLid* is then popped, leaving just the five features as shown in *Fig.30* below. Afterwards, the date feature is then set as the index.

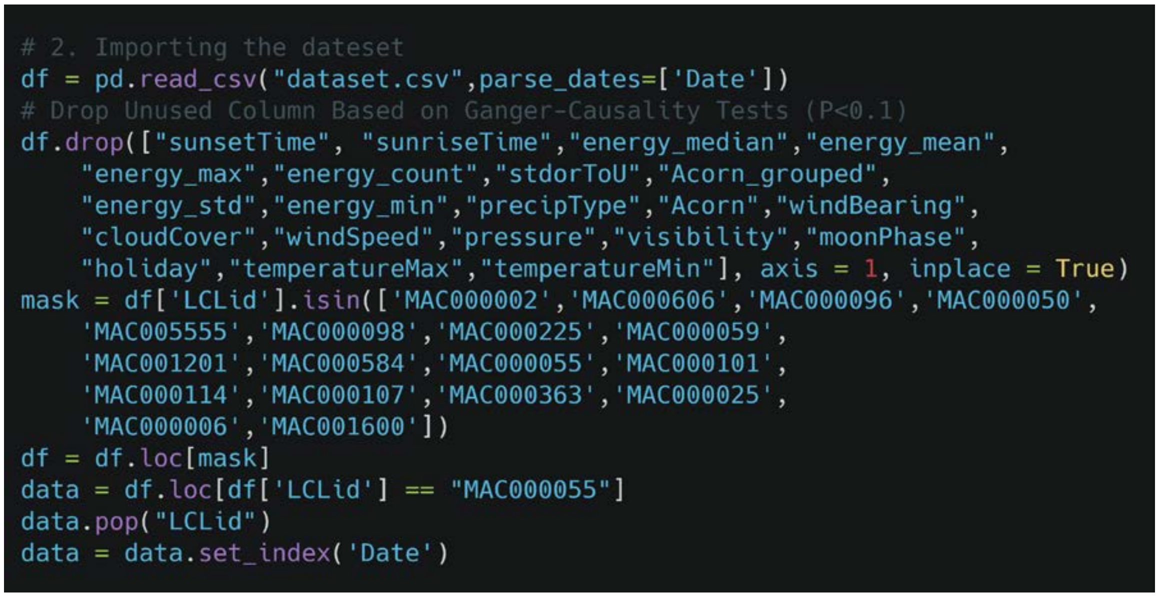

```
# 2. Importing the dateset
df = pd.read_csv("dataset.csv",parse_dates=['Date'])
# Drop Unused Column Based on Ganger-Causality Tests (P<0.1)
df.drop(["sunsetTime", "sunriseTime","energy_median","energy_mean",
    "energy_max","energy_count","stdorToU","Acorn_grouped",
    "energy_std","energy_min","precipType","Acorn","windBearing",
    "cloudCover","windSpeed","pressure","visibility","moonPhase",
    "holiday","temperatureMax","temperatureMin"], axis = 1, inplace = True)
mask = df['LCLid'].isin(['MAC000002','MAC000606','MAC000096','MAC000050',
    'MAC005555','MAC000098','MAC000225','MAC000059',
    'MAC001201','MAC000584','MAC000055','MAC000101',
    'MAC000114','MAC000107','MAC000363','MAC000025',
    'MAC000006','MAC001600'])
df = df.loc[mask]
data = df.loc[df['LCLid'] == "MAC000055"]
data.pop("LCLid")
data = data.set_index('Date')
```

Fig. 30 - Importing the dataset (VAR)

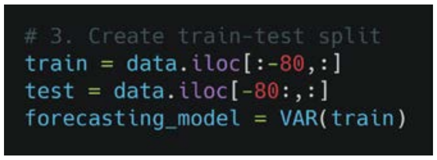

```
# 3. Create train-test split
train = data.iloc[:-80,:]
test = data.iloc[-80:,:]
forecasting_model = VAR(train)
```

Fig. 31 - Create train-test split (VAR)

After importing the dataset and converting it into a dataframe, a train-test split is then created. The last 80 days of the two year daily data will be the test data, and the rest will be training data. This equates to ±90% train data and ±10% test data as shown in *Fig. 31* above.

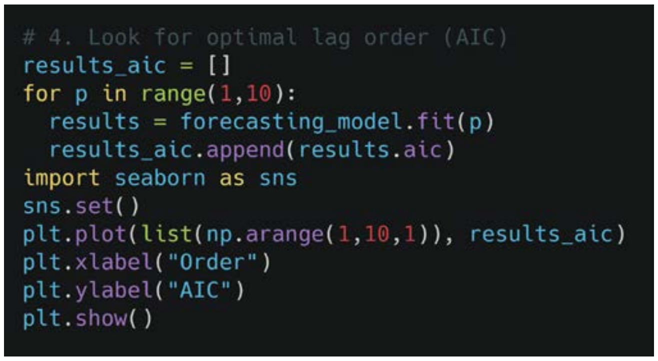

```
# 4. Look for optimal lag order (AIC)
results_aic = []
for p in range(1,10):
  results = forecasting_model.fit(p)
  results_aic.append(results.aic)
import seaborn as sns
sns.set()
plt.plot(list(np.arange(1,10,1)), results_aic)
plt.xlabel("Order")
plt.ylabel("AIC")
plt.show()
```

Fig. 32 - AIC test for optimal lag order (VAR)

An Aikaike's information criterion or AIC test for short, was performed to check for the lag order for the model to be fitted with. The maximum lag order for the AIC test was set at 10 as shown in *Fig. 32* above, and they were plotted as seen in *Fig. 33* below. It is subjective to a certain extent, as what we are looking for is the lowest dip in value with a high gradient (steep) increase immediately afterwards. In this case, the lag order is 6.

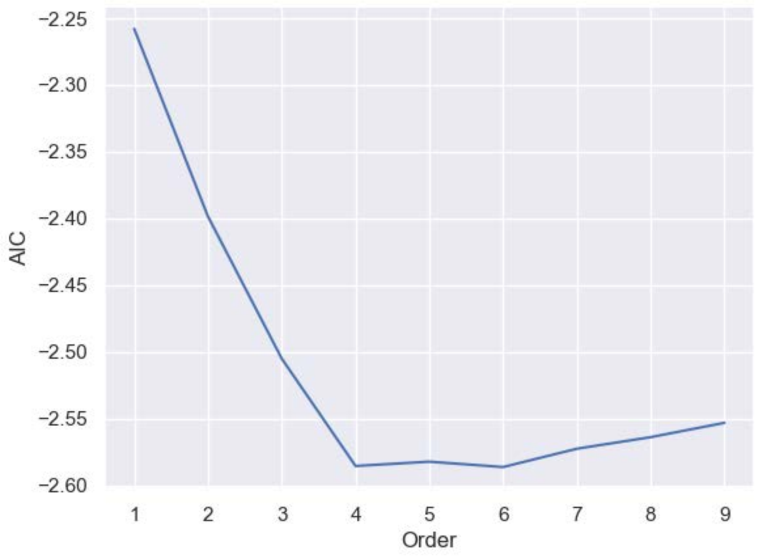
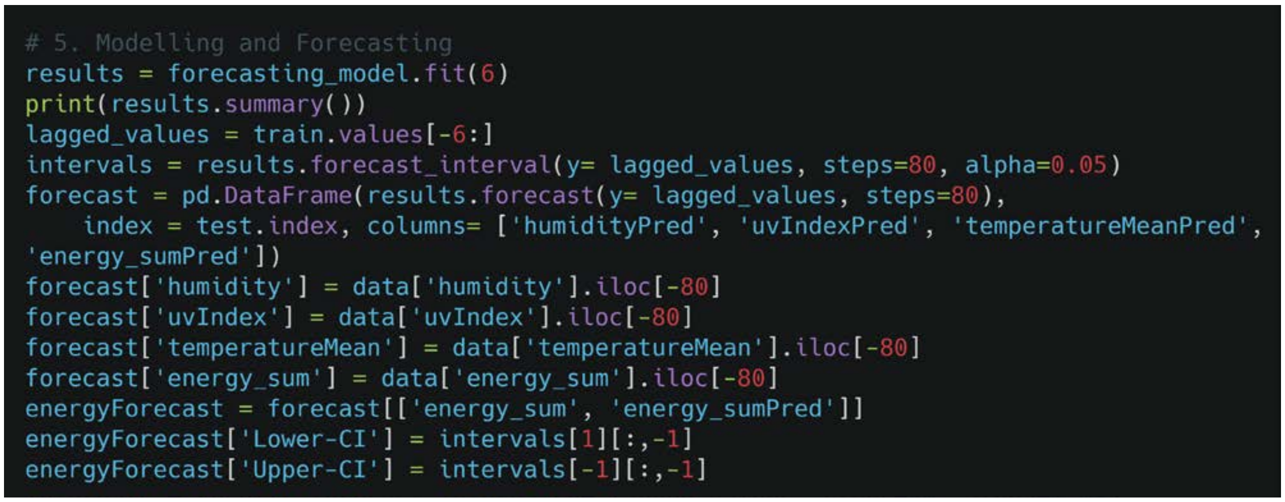


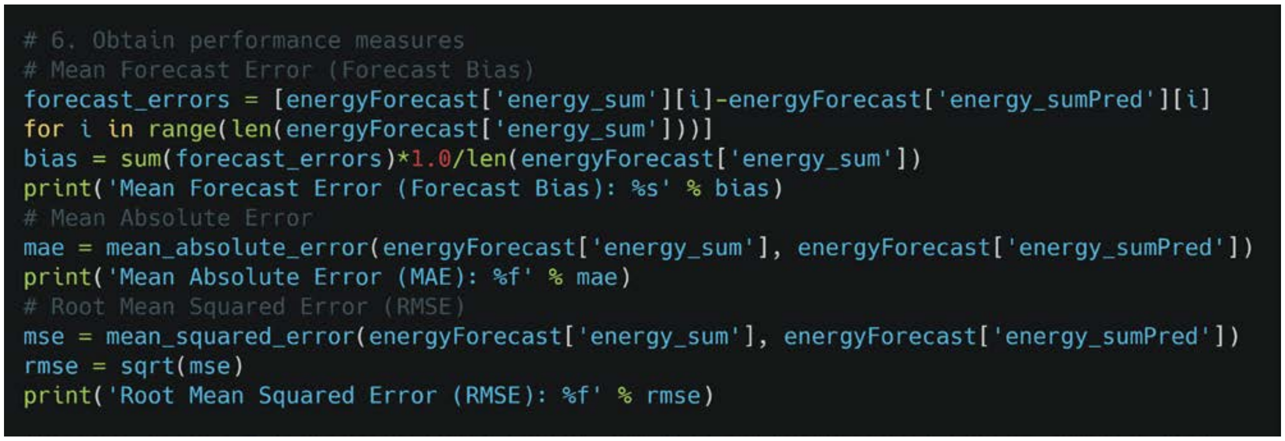
Fig. 33 – AIC test (VAR)



```
# 5. Modelling and Forecasting
results = forecasting_model.fit(6)
print(results.summary())
lagged_values = train.values[-6:]
intervals = results.forecast_interval(y= lagged_values, steps=80, alpha=0.05)
forecast = pd.DataFrame(results.forecast(y= lagged_values, steps=80),
    index = test.index, columns= ['humidityPred', 'uvIndexPred', 'temperatureMeanPred',
'energy_sumPred'])
forecast['humidity'] = data['humidity'].iloc[-80]
forecast['uvIndex'] = data['uvIndex'].iloc[-80]
forecast['temperatureMean'] = data['temperatureMean'].iloc[-80]
forecast['energy_sum'] = data['energy_sum'].iloc[-80]
energyForecast = forecast[['energy_sum', 'energy_sumPred']]
energyForecast['Lower-CI'] = intervals[1][:,-1]
energyForecast['Upper-CI'] = intervals[-1][:,-1]
```

Fig. 34 - Modelling and forecasting (VAR)

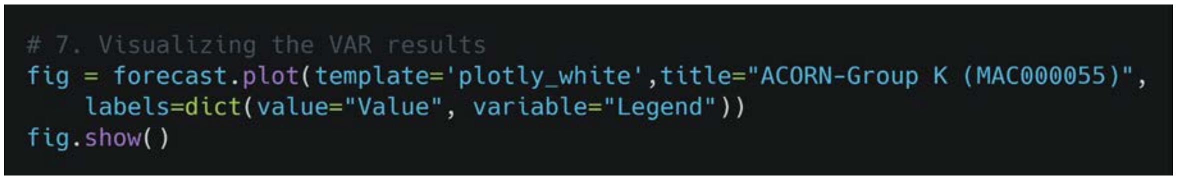
We can then proceed to fitting the model by calling the *.fit()* function and passing the number 6 as the parameter. Printing the summarized data of the fitted model by calling the *.summary()* function. Manually lagging the values in the *train* Dataframe is also necessary, in this case it was passed through a variable called *lagged_values*. Confidence intervals for the forecast can also be obtained by calling the *.forecast_interval()* function located, shown on the 5th line of the code snippet. Forecasting can now be executed by calling the *.forecast()* function, which *takes lag order* and *steps* as the required parameters. Parameter *alpha* was set to 0.05 for a consistent value throughout all the forecast of different households as shown in *Fig. 34* above.



```
# 6. Obtain performance measures
# Mean Forecast Error (Forecast Bias)
forecast_errors = [energyForecast['energy_sum'][i]-energyForecast['energy_sumPred'][i]
for i in range(len(energyForecast['energy_sum']))]
bias = sum(forecast_errors)*1.0/len(energyForecast['energy_sum'])
print('Mean Forecast Error (Forecast Bias): %s' % bias)
# Mean Absolute Error
mae = mean_absolute_error(energyForecast['energy_sum'], energyForecast['energy_sumPred'])
print('Mean Absolute Error (MAE): %f' % mae)
# Root Mean Squared Error (RMSE)
mse = mean_squared_error(energyForecast['energy_sum'], energyForecast['energy_sumPred'])
rmse = sqrt(mse)
print('Root Mean Squared Error (RMSE): %f' % rmse)
```

Fig. 35 – Obtaining performance measures (VAR)

For the purpose of evaluation (see chapter 6), mean forecast error, MAE and RMSE is obtained by calling the *mean_absolute_eror()* and *mean_squared_error()* function respectively, which takes in the actual value and predicted value as the parameters as shown in *Fig. 35* above.



```
# 7. Visualizing the VAR results
fig = forecast.plot(template='plotly_white',title="ACORN-Group K (MAC000055)",
    labels=dict(value="Value", variable="Legend"))
fig.show()
```

Fig. 36 - Plotting Whole Time Series Result (VAR)

The predicted results can now be forecasted in a time-series graph. The syntax is slightly different, as a Dataframe is directly plotted instead of arrays as shown in *Fig. 36* above. The result of the plot is as shown in *Fig. 38* below. The purple-coloured line is the predicted energy consumption for household MAC000055, and the light, green-coloured line represents the actual energy consumption. It is seen that the model over-predicted the energy consumption by about 1.5 KWh.

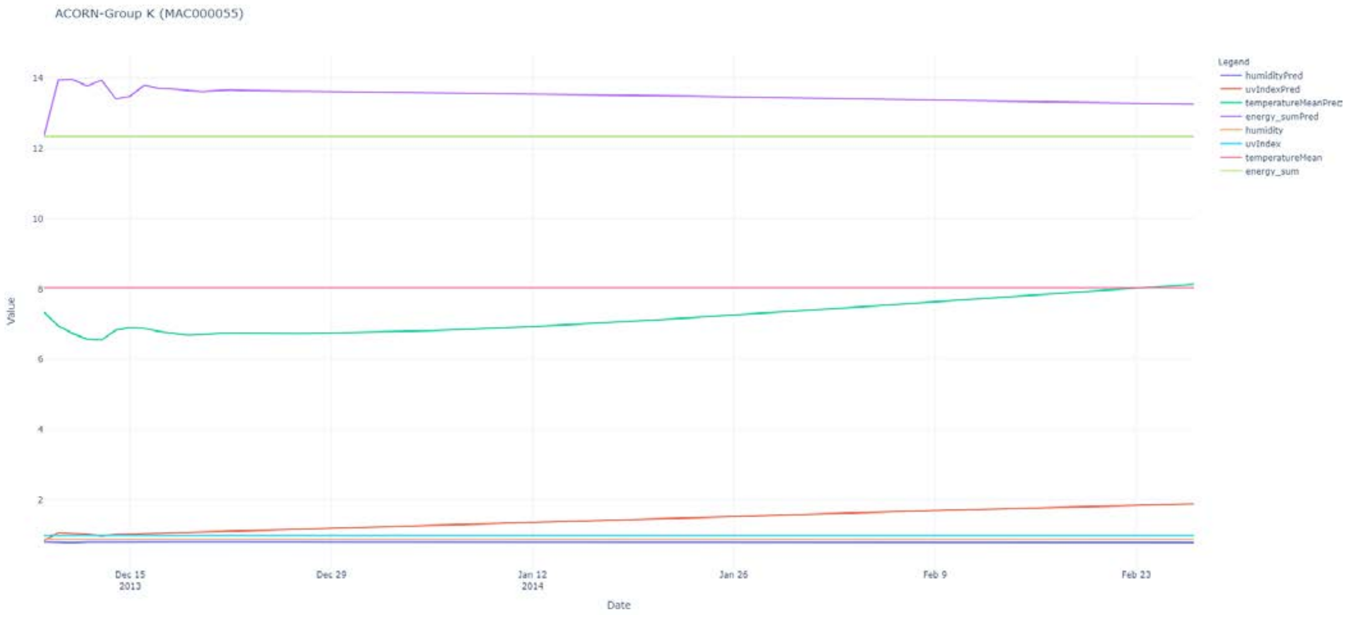
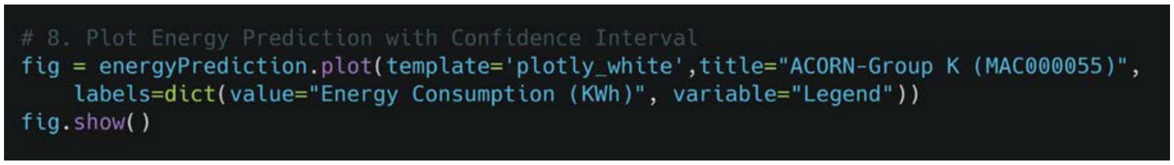


Fig. 38 - All time-series forecast plotted (VAR)



```
# 8. Plot Energy Prediction with Confidence Interval
fig = energyPrediction.plot(template='plotly_white',title="ACORN-Group K (MAC000055)",
    labels=dict(value="Energy Consumption (KWh)", variable="Legend"))
fig.show()
```

Fig. 37 - Plotting energy prediction with confidence interval (VAR)

To view and further analyse just the energy consumption prediction, we can use the code as seen in *Fig. 37* above. The result of the plot is as shown in *Fig. 39* below. The blue- line in *Fig.39* is the actual energy consumption and the red-coloured line is the predicted energy consumption. The purple-coloured line represents the upper confidence interval, while the green-coloured line represents the lower confidence interval.

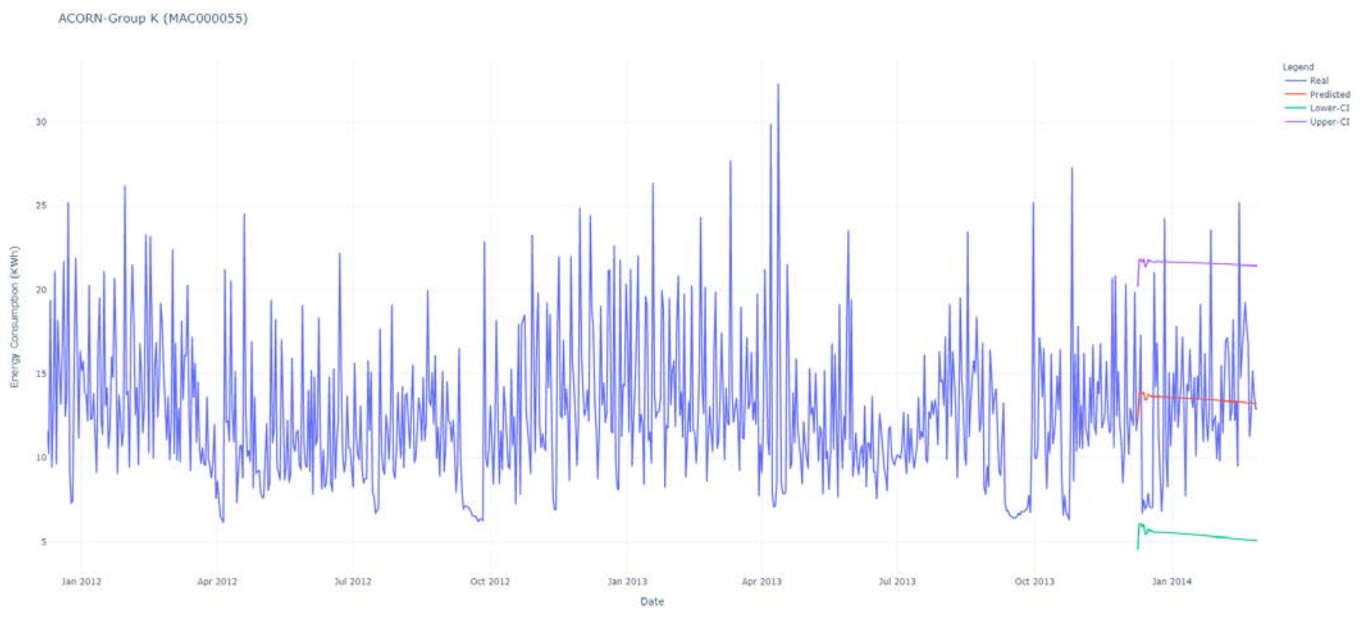
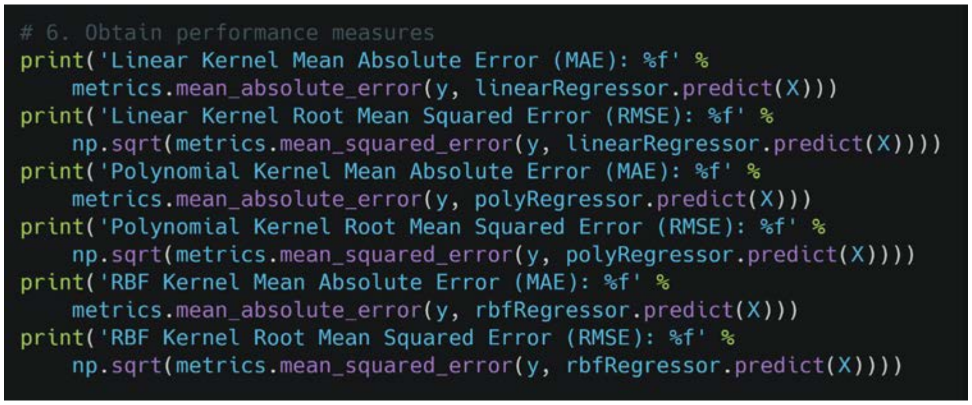


Fig. 39 - Detailed plot with confidence intervals (VAR)

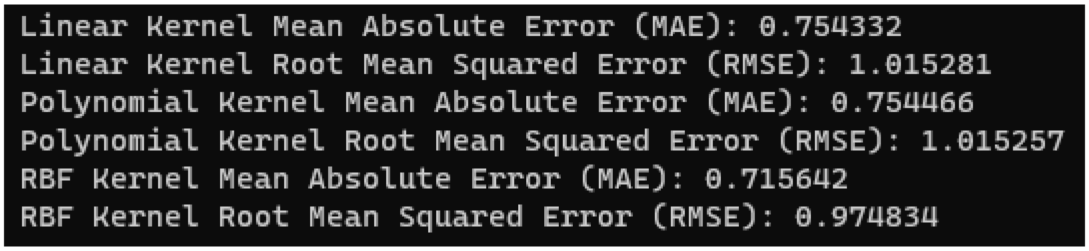
5. Evaluation

From visual checks, both SVR (seen in *Fig. 28*) and VAR (seen in *Fig. 39*) time series forecasts provide a relatively smooth prediction line. This is important, as smooth prediction lines ignores outliers and help predict trends better. As seen in *Fig. 40* the model with linear and polynomial regressors failed to predict trends in energy consumption. However, the model with the RBF kernel seems to be able to predict trends and create a smooth prediction line without the need of data smoothing.



```
# 6. Obtain performance measures
print('Linear Kernel Mean Absolute Error (MAE): %f' %
    metrics.mean_absolute_error(y, linearRegressor.predict(X)))
print('Linear Kernel Root Mean Squared Error (RMSE): %f' %
    np.sqrt(metrics.mean_squared_error(y, linearRegressor.predict(X))))
print('Polynomial Kernel Mean Absolute Error (MAE): %f' %
    metrics.mean_absolute_error(y, polyRegressor.predict(X)))
print('Polynomial Kernel Root Mean Squared Error (RMSE): %f' %
    np.sqrt(metrics.mean_squared_error(y, polyRegressor.predict(X))))
print('RBF Kernel Mean Absolute Error (MAE): %f' %
    metrics.mean_absolute_error(y, rbfRegressor.predict(X)))
print('RBF Kernel Root Mean Squared Error (RMSE): %f' %
    np.sqrt(metrics.mean_squared_error(y, rbfRegressor.predict(X))))
```

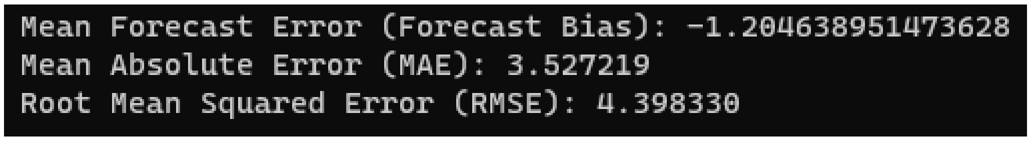
Fig. 40 - Code for SVR performance measures



```
Linear Kernel Mean Absolute Error (MAE): 0.754332
Linear Kernel Root Mean Squared Error (RMSE): 1.015281
Polynomial Kernel Mean Absolute Error (MAE): 0.754466
Polynomial Kernel Root Mean Squared Error (RMSE): 1.015257
RBF Kernel Mean Absolute Error (MAE): 0.715642
RBF Kernel Root Mean Squared Error (RMSE): 0.974834
```

Fig. 41 - SVR Performance Measure

The code for obtaining performance measures for the SVR model are as shown in *Fig. 40*. The results as seen in *Fig. 41*, supports the hypothesis that the model with the RBF kernel has the most accurate forecast. Both its MAE and RMSE are the lowest compared to the other models including the VAR model. The results for the VAR model are as shown in *Fig. 42* below.



```
Mean Forecast Error (Forecast Bias): -1.204638951473628
Mean Absolute Error (MAE): 3.527219
Root Mean Squared Error (RMSE): 4.398330
```

Fig. 42 - VAR Performance Measure

The MAE and RMSE value of the SVR model are superior to the VAR model. The forecast bias for the VAR model is a negative value, meaning that the model under predicted the energy consumption.

6. Summary and Reflection

This section covers all the work that has been done as well as the reflections. After further research on related works, viable algorithms, and models, I found that SVR yields a significantly better result in comparison to VAR when it comes to time series forecasting accuracy. However, by the looks of it, SVR is better for long term forecasting, due to its ability to create a smooth trend. That being said, VAR had the edge over short term time series forecasting.

6.1. Project Management

This project was designed to be completed sequentially, with all the data collection and refinements done first prior to designing the model and implementing them. The reason for this style of progress is to make sure that the

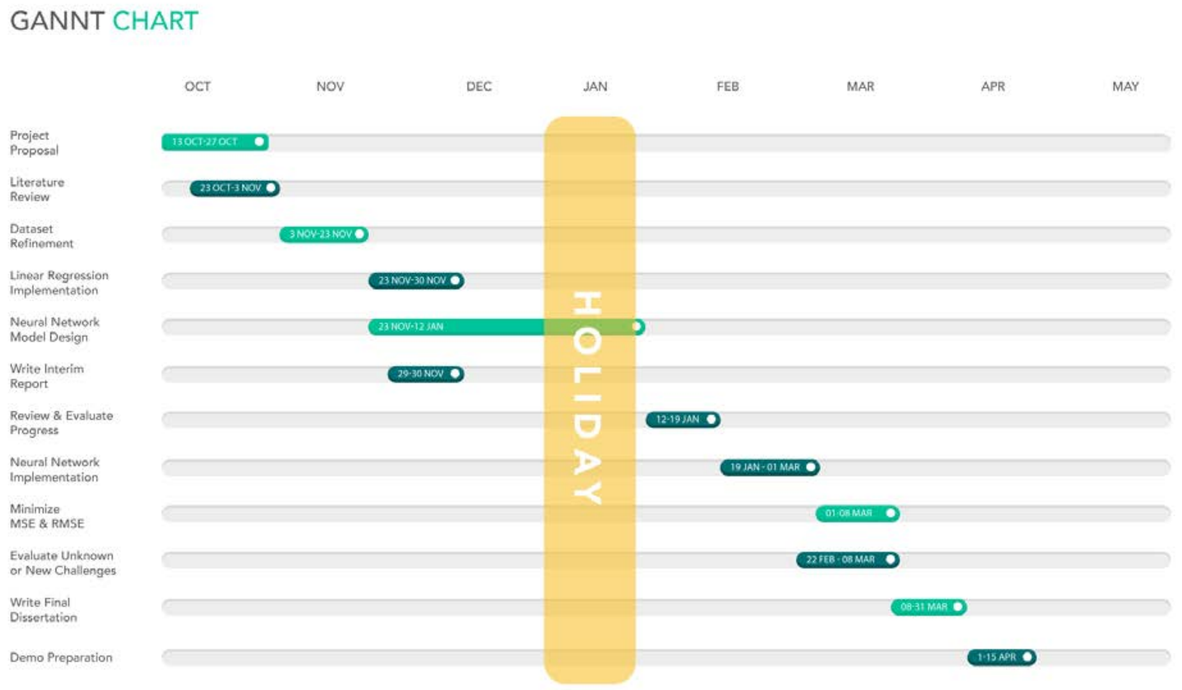


Fig. 43 – Original Gannt Chart

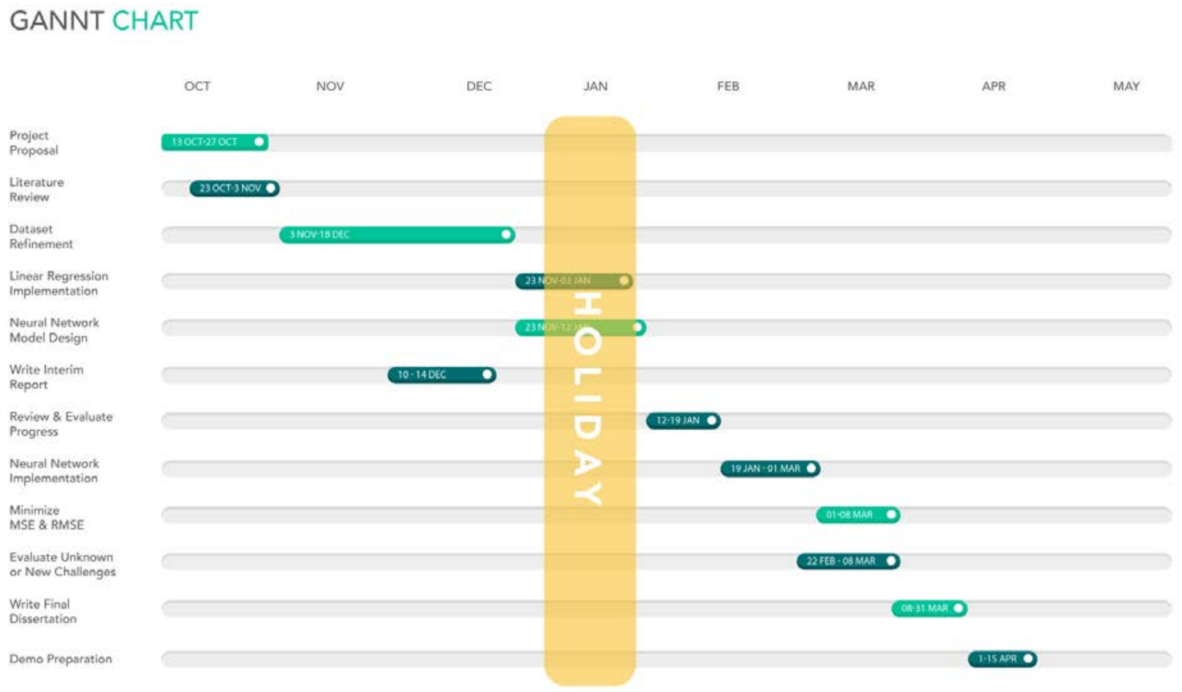


Fig. 44 – Final Gannt Chart

model design and implementation run as smooth as possible, and to also maintain consistencies in the implementation stage later on. By refining the datasets first, it will rule them out of any unforeseeable bugs and/or errors. *Fig. 43* was the original Gantt chart in the beginning of the project. However, it became apparent that refining the data would take longer and I found new model designs and algorithms from related works that would significantly improve the overall quality of this project. Thus, I have adjusted the project timeline as shown in *Fig. 44***.**

6.2. Contributions and Reflection

Working on this project, I have encountered some time management issues due to the unexpected load of refining the dataset and the switching of algorithms from support vector regression (SVR) to include vector autoregression (VAR) as well. I chose to move forward with the new idea after much consideration, as it would improve the quality and result of my final dissertation and show another perspective on traditional econometric and machine learning models.

Since a few changes were made, I needed to update my Gantt chart to adjust my schedule to it. This includes adding a new algorithm, VAR on top of SVR as a comparison and modify the feature selection process for a consistent dataset across the algorithms.

During my literature review, I came up with ideas coincidently changed a few of my key objectives. Instead of just implementing one machine learning algorithm, I thought it would be better to compare it with a traditional econometric model.

Although it was a slow start, I am happy with my dissertation. There are things that can be improve. If I were to do it again, I would do a variety of train-test splits for each algorithm, as well as more sample size. Since econometric models are sensitive with noise, in hindsight I should have cleaned up the data more and perform more tests, not just AIC to determine the lag order. That being said, reviewing related works also helped me understand more about the current industry practices in modelling for time series problems and how I can contribute to the community.

7. Bibliography

sklearn.svm.SVR. Retrieved from URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

(2014). SmartMeter Energy Consumption Data in London Households. Retrieved from URL: https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households

Bhattacharyya S., Timilsina G. (2009). A Comparative Study of Energy Demand Models. The World Bank Policy Research Working Paper.

Sreenivasa, S. (October 12). Radial Basis Function (RBF) Kernel: The Go-To Kernel. Retrieved from URL: https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a#:~:text=RBF%20Kernel%20is%20popular%20because,and%20not%20the%20entire%20dataset.

What is ACORN. Retrieved from URL: https://acorn.caci.co.uk/what-is-acorn

Guillard, E. (2020, April 22). Forecasting Energy Consumption using Machine Learning and AI. Retrieved from URL: https://www.dexma.com/blog-en/forecasting-energy-consumption-using-machine-learning-and-ai/

Energy Efficiency in London: How energy efficiency benefits residents and business in London. Association for the Conservation of Energy. Retrieved from URL: https://www.theade.co.uk/assets/docs/resources/Energy-Efficiency-in-London.pdf

Generating Electricity. Retrieved from URL: https://www.bbc.co.uk/bitesize/guides/z3qd7p3/revision/10

Luoma, J. (2009, July 13). The Challenge for Green Energy: How to Store Excess Electricity. Retrieved from URL: https://e360.yale.edu/features/the_challenge_for_green_energy_how_to_store_excess_electricity

(2019, February 22). Fact Sheet | Energy Storage (2019). https://www.eesi.org/papers/view/energy-storage-2019

Alam, M. (2020, April 4). Multivariate time series forecasting. Retrieved from URL: https://towardsdatascience.com/multivariate-time-series-forecasting-653372b3db36

Dhiraj, K. (2020, March 6). Support Vector Regression in Python Using Scikit-Learn. Retrieved from URL: https://heartbeat.fritz.ai/support-vector-regression-in-python-using-scikit-learn-89cc18e933b7

Jean-Michel, D. (2017, October 20). Make a forecast system of the French national energy consumption. Retrieved from URL: https://towardsdatascience.com/make-a-forecast-system-of-the-french-national-energy-consumption-4f946b91381b

Maitra, S. (2020, June 18). Retrieved from URL: https://towardsdatascience.com/vector-autoregressive-for-forecasting-time-series-a60e6f168c70

Disci, S. (2020, September 29). Time Series Forecasting: KNN vs. ARIMA. Retrieved from URL: https://www.r-bloggers.com/2020/09/time-series-forecasting-knn-vs-arima/

Kim, E. (2017, December 20). Everything You Wanted to Know about the Kernel Trick. Retrieved from URL: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Raj, A. (2020, October 3). Unlocking the True Power of Support Vector Regression. Retrieved from URL: https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0

Mean Absolute Percentage Error. Retrieved from URL: https://www.statisticshowto.com/mean-absolute-percentage-error-mape/

Prabhakaran, S. Vector Autoregression (VAR) – Comprehensive Guide with Examples in Python. Retrieved from URL: https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/

Khair, U., Fahmi, H., Hakim, S., Rahim, R. (2017), Forecasting Error Calculation with Mean Absolute Deviation and Mean Absolute Percentage Error. IOP Conf. Series: Journal of Physics: Conf. Series 930 (2017) 012002. Retrieved from URL: https://iopscience.iop.org/article/10.1088/1742-6596/930/1/012002/pdf#:~:text=Mean%20Absolute%20Percentage%20Error%20Mean,are%20evident%20for%20that%20period.

Granger Causality. Retrieved from URL: https://www.real-statistics.com/time-series-analysis/time-series-miscellaneous/granger-causality

G. Peter Zhang. (2003), Time series forecasting using a hybrid ARIMA

and neural network model, Neurocomputing, 50, 159-157.

G. Peter Zhang. (2001), An investigation of neural networks for linear time-series

Forecasting. Computers & Operations Research, 28, 1183-1202.

Le, T., Vo, M., Hwang, E., Rho, S., Baik, S. (2019). Improving Electric Energy Consumption Prediction Using CNN and Bi-LSTM, applied sciences.

Vapnik, V., Cortes, C. (1995). Support-Vector Networks, Machine Learning, 20, 273-279

Livieris1, I., Pintelas1, E., Pintelas, P. (2020). A CNN–LSTM model for gold price time-series forecasting.