ECE9603/9063: Data Analytics Foundations

by

Omar Awadallah

oawadall@uwo.ca

Instructor:

Dr. Katarina Grolinger

Department of Electrical and Computer Engineering

Western University

London, Ontario, Canada

Table of Contents

1. Problem Description....................2

2. Dataset Description....................2

2.1 General Overview....................2

2.2 Data Pre-Processing....................3

3. Background of Algorithms....................6

3.1 Seasonal ARIMA....................6

3.2 Support Vector Regression....................7

3.3 XGBoost....................8

4. Methodology....................8

4.1 Application of Algorithms....................8

4.2 Evaluation Procedure....................9

5. Results and Discussions....................10

6. References....................13

1. Problem Description

The importance of cars and automobiles evolves continuously as the economy grows and job opportunities increase. Cars provide more freedom for mobility and significantly decrease travel times. Therefore, car sales are important as demand increases and these sales are often an indicator of the economic growth in countries.

The forecasting problem that is discussed in this report aims to forecast the future quantity of cars sold every month in the country of Norway. Forecasting the future quantities of cars sold every month helps businesses to manage their resources and adjust their plans for cash flow management, it also helps governments to analyze changes in the economy. A dataset that contains information about the number of different car models sold every month for ten years (2007-2017) in Norway is used to solve this forecasting problem, the information about this dataset is discussed in section 2 of this report.

2. Dataset Description

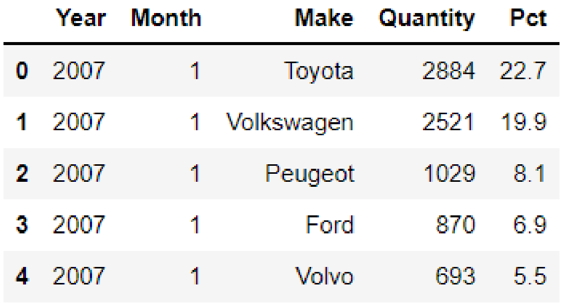
2.1 General Overview

The dataset that is used to forecast the total monthly quantities of cars sold in Norway is obtained from Kaggle [1] and is collected by the Norwegian road association. The dataset is time-series and contains five different attributes; the year which ranges from 2007 to 2017, the month, which is a value from 1 to 12, the car makes, which contains 38 different types of cars (e.g., Toyota, Ford, Volvo), the quantity sold for each car make every month, and the percent share in monthly total. The dataset consists of five columns corresponding to each attribute, and 4377 rows where each row represents monthly quantities sold for a specific car make and their percent share. Table 1 shows a summary of the dataset attributes, and Figure 1 shows the first five rows of the dataset.

Table 1: Overview of initial data attributes

	Description	Type	Values
Year	The year when the sales occurred.	Integer	January 2007 to January 2017.
Month	The month when the sales occurred.	Integer	1 to 12.
Make	The make of the car or the car's name.	String	38 different car makes (e.g., Toyota, Ford, Volvo).

Quantity	The total number of cars sold in each month by make.	Integer	1 to 3017.
Monthly Percent Share (Pct)	The monthly percent share for each car make sold.	Float	Equal to or greater than 0.0.

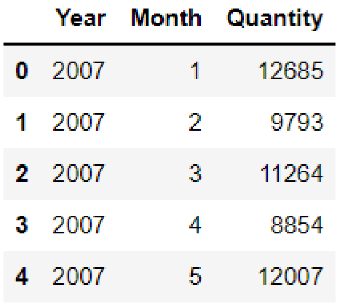


	Year	Month	Make	Quantity	Pct
0	2007	1	Toyota	2884	22.7
1	2007	1	Volkswagen	2521	19.9
2	2007	1	Peugeot	1029	8.1
3	2007	1	Ford	870	6.9
4	2007	1	Volvo	693	5.5

Figure 1. Initial dataset

2.2 Data Pre-Processing

As mentioned in section 1, the goal is to predict the monthly total quantity of cars sold, this means all car makes must be grouped to represent the total quantities sold every month. Therefore, the total quantity of all cars does not depend on the car make and the *Make* attribute is dropped from the data. Because the car make is dropped, the percent share for each car make is also dropped. Only the *Year*, *Month*, and *Quantity* attributes are used for training and forecasting. The next step is to group all quantities sold by the year and month and sum them up, this results in a dataset of three columns (*Year, Month, Quantity)*, and a total of 121 rows (10 years*12 months + the first month in 2017). Figures 2 and 3 show the dataset after filtering and a plot of the total number of cars sold each month for 10 years respectively.



	Year	Month	Quantity
0	2007	1	12685
1	2007	2	9793
2	2007	3	11264
3	2007	4	8854
4	2007	5	12007

Figure 2. Filtered dataset

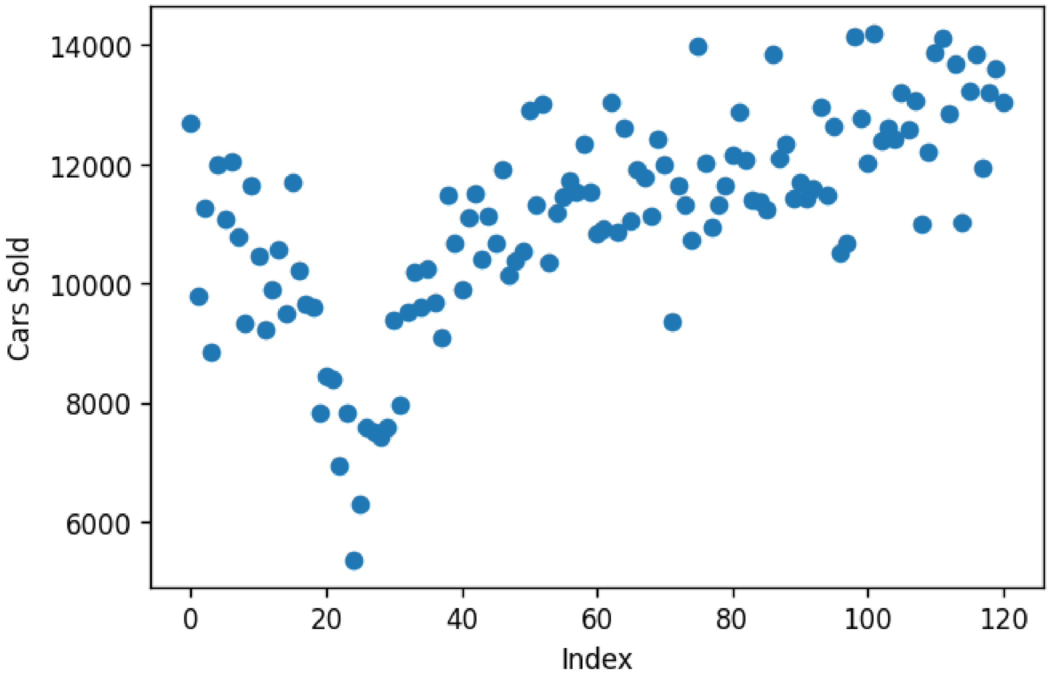


Figure 3. A plot of monthly total cars sold for 10 years

Since this is time-series data, it is important to extract the trend and seasonality plots, these plots decide whether further data pre-processing is needed or not, and it also helps in algorithm selection. Figure 4 shows the observed, trend, seasonal and residual plots of the filtered data. The figure shows that there is an increasing trend in general, and the quantity of cars sold tends to increase over the months. However, there is a significant decrease in the number of cars sold between 2008 and 2009, this is the same period as the great recession that started in 2008 which explains why the trend drops. Therefore, data normalization is needed to reduce the effect of these outliers. In the seasonality plot, there is seasonality in the data because a fixed period of one year is observed. Also, the seasonality plot shows that at the beginning of a new year there is always a significant drop in the number of cars sold, this is most likely due to holidays because the holiday season is always around the new year, and almost no car sales are made during holidays. Therefore, the seasonality factor needs to be accounted for when choosing algorithms.

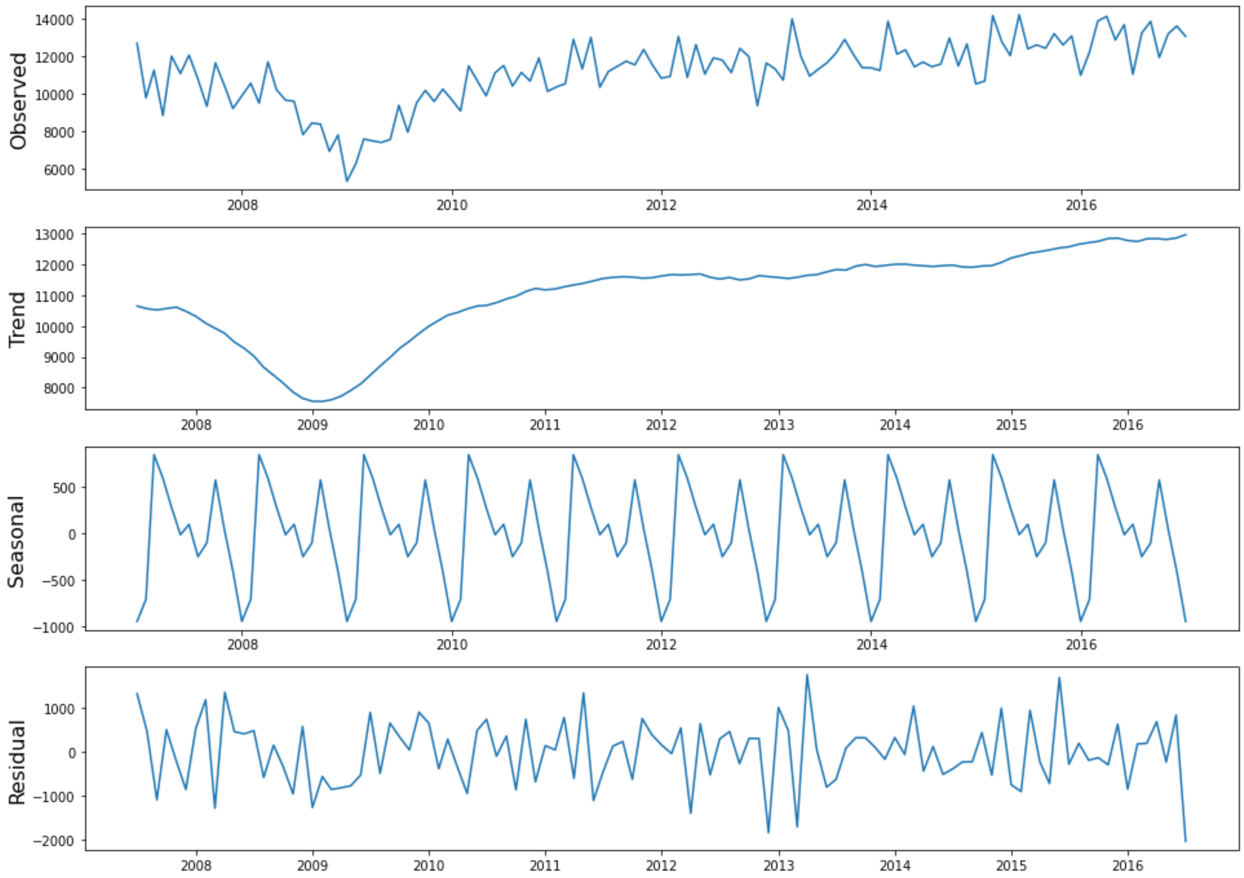
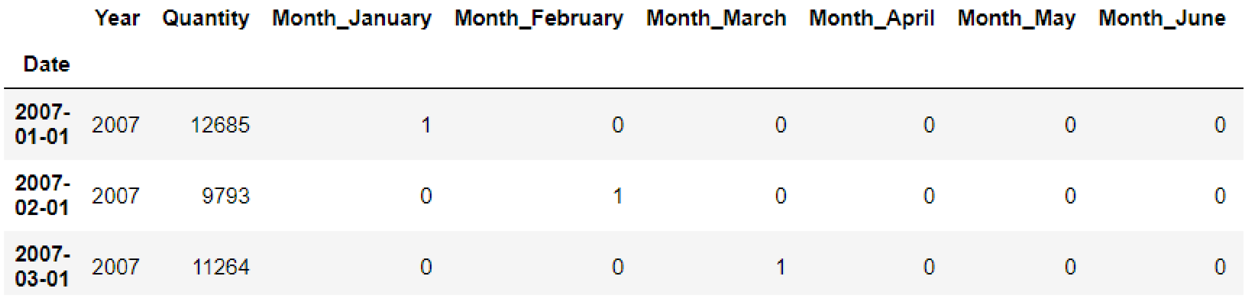


Figure 4. Observed, Trend, Seasonal, and Residual plots of the filtered dataset

The last step of data preparation is transforming the *Month* attribute using one-hot encoding. The months range from 1 to 12, where the difference between each month and the next is only one, except for the difference between December and January, where the algorithm interprets the difference between them to be 11 (12-1), this causes the algorithm to weigh these two months more heavily than others. Using one-hot encoding, the algorithm can achieve a correct understanding of the data used. Figure 5 shows the final dataset used in training which contains 14 columns (attributes) and 121 rows in total.



	Year	Quantity	Month_January	Month_February	Month_March	Month_April	Month_May	Month_June
Date								
2007-01-01	2007	12685	1	0	0	0	0	0
2007-02-01	2007	9793	0	1	0	0	0	0
2007-03-01	2007	11264	0	0	1	0	0	0

Figure 5. Final filtered dataset

Before the start of the training process, the data is split into training and testing sets with a ratio of 8:2, the training set consists of 96 samples and ranges from January 2007 to December 2014, while the test set consists of 25 samples and ranges from January 2015 to January 2017. The data is normalized with *z-normalization* that is described in Equation 1, where *x* is the actual value, μ is the training mean and σ is the training standard deviation.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

3. Background of Algorithms

3.1 Seasonal ARIMA

The Seasonal ARIMA model or SARIMA (Seasonal AutoRegressive Integrated Moving Average) is similar to the ARIMA model that is an improvement to the original ARMA model. The ARMA model is a time-series model for stationary data, it includes only two elements, the trend autoregressive order *(p)* and the trend moving average order *(q)*, in the form of *ARMA(p,q)*. For non-stationary data, the ARIMA model is often used which is similar to ARMA but adds a third element, the trend differencing order *(d)*, in the form of *ARIMA(p,d,q)*. The equation for an ARIMA model is shown in Equation 2.

$$\left(1 - \varphi_1 B - \cdots - \varphi_p B^p\right)(1 - B^d) y_t = c + \left(1 + \theta_1 B + \cdots + \theta_q B^q\right) e_t \tag{2}$$

Where:

- B, backshift operator
- y_t, is the original target value
- c, is a constant value
- e_t, is the error
- $\varphi \; and \; \theta,$ coefficients

SARIMA builds on the ARIMA model and adds a seasonal component to account for any seasonality in the data. SARIMA contains the original trend elements and four additional seasonal elements, seasonal autoregressive order *(P)*, seasonal differencing order *(D)*, seasonal moving average order *(Q)*, and the frequency for one seasonal period *(m)*, in the form of *SARIMA(p,d,q)(P,D,Q,m)*.

3.2 Support Vector Regression

The Support Vector Regression (SVR) model is a learning algorithm that uses supervised data which means that it implements supervised learning. SVR is similar to Support Vector Machines (SVM) as both require the use of kernels that decide whether the algorithm is linear or non-linear. Types of kernels include linear, polynomial, hyperbolic, and Gaussian kernels. However, SVR differs from SVM as it is used for regression problems to predict values, while SVM is used for classification problems to predict types or classes. SVR uses hyperplanes which are boundaries that the algorithm uses to predict targets, these hyperplanes are defined by support vectors. The support vectors are obtained by minimizing:

$$\frac{1}{2}\|w\|^2 \tag{3}$$

For linear SVR with constraints:

$$y_i - wx_i - b \ \leq \ \epsilon \quad \boldsymbol{and} \quad wx_i + b - \ y_i \leq \epsilon \tag{4}$$

where $wx_i + b$ is the prediction, y_i are the actual target value and ϵ is the threshold value for accepted errors. Figure 6 shows an example of linear SVR where hyperplane boundaries are defined using support vectors.

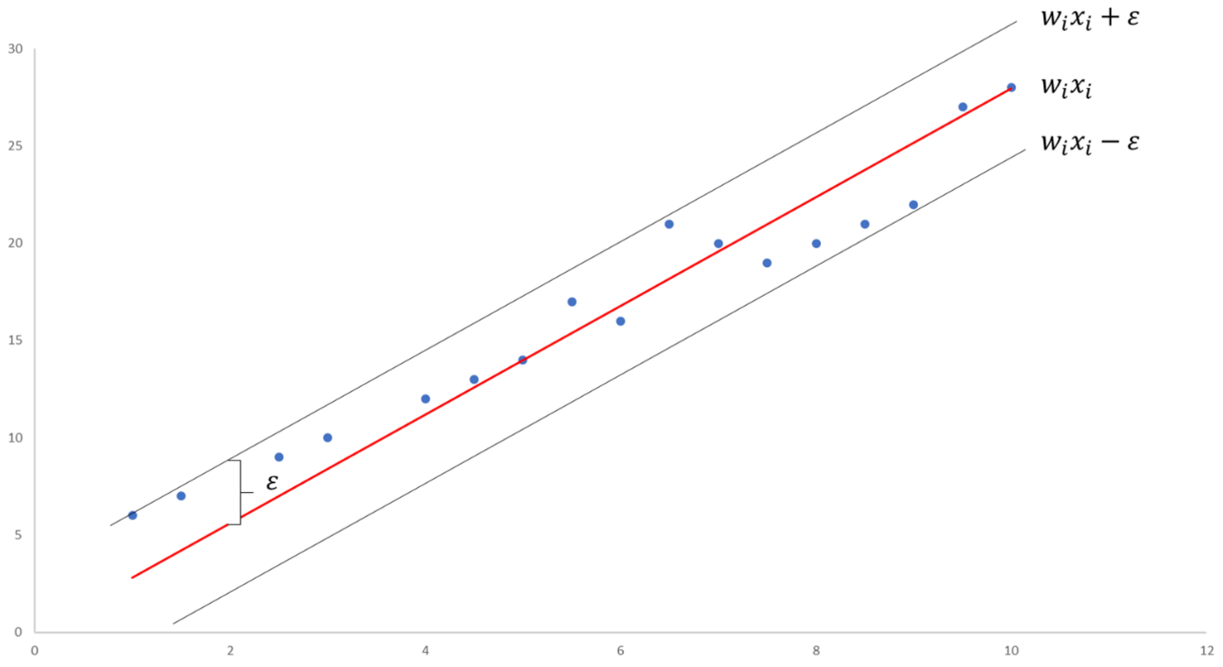


Figure 6. SVR hyperplane for data points [2]

3.3 XGBoost

XGBoost is a supervised learning algorithm that is used for both classification and regression problems, and it implements gradient-boosting decision trees, it is called Extreme Gradient Boosting because of its performance as it is ten times faster than other algorithms [3]. XGBoost relies on gradient descent algorithms, and it offers three different gradient boosting techniques, regular gradient boosting with learning rates, stochastic gradient boosting, and regularized gradient boosting with L1 (lasso regression) and L2 (ridge regression). XGBoost uses multiple decision trees and depends on weights that are assigned to the inputs, which are then provided to the decision trees to generate predictions. The weights of the wrong predictions are increased and fed again to the next decision tree to improve predictions. XGBoost computes the optimal weights and their corresponding optimal value using Equations 5 and 6 respectively [3].

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{5}$$

$$obj_j^* = -\frac{1}{2}\sum_{1}^{T}\frac{{G_j}^2}{H_j + \lambda} + \gamma T \tag{6}$$

Where:

- G and H, gradient statistics
- $\lambda,$ regularization parameter (L1/L2)
- $\gamma,$ pruning parameter
- $T,$ the number of leaves in the decision tree.

4. Methodology

4.1 Application of Algorithms

As mentioned in section 2.2, the data is split into training and testing sets. The normalized training and testing sets are split into normalized inputs and targets; *Xtrain_norm* and *Xtest_norm* which contain the *Year* attribute and the 12 features that represent the one-hot encoding for the initial *Month* attribute, and *ytrain_norm* and *ytest_norm* which contain the *Quantity* attribute. For SARIMA, only the normalized *Quantity* attribute in the training set is used to fit the model, while the predictions on the training and testing sets are obtained by predicting the training and testing date ranges; for the training set, it ranges from January 2007 to December 2014, for the testing set it ranges from January 2015 to January 2017. For SVR and XGBoost, *Xtrain_norm* and *ytrain_norm* are used to fit the models, while the predictions on the training and testing sets are obtained by predicting on *Xtrain_norm* and *Xtest_norm* respectively. Once predictions are obtained, they are denormalized using Equation 7 to get the real prediction values, the training mean and standard deviation values that are used to normalize the data are used again to reverse the normalization. Equation 7 is derived by solving for *x* in Equation 1.

$$x = z\sigma + \mu \qquad (7)$$

The default parameters for SARIMA do not take seasonality into consideration, the default parameters are *(p,d,q)(P,D,Q,m) = (1,0,0)(0,0,0,0)*. Therefore, the value of P is set to one to include the seasonality component, and the frequency *m* is set to 12 because in a single period of one year 12 samples represent each month in the year. Also, d and q are randomly set to 1 to achieve a response on the testing set. The final configuration is *SARIMA(1,1,1)(1,0,0,12)*. For SVR, default configurations did not produce any response on both the training and testing sets, hence the default kernel of *rbf* is set to *poly* (polynomial kernel), and randomly chosen values of 0.7, 50, and 0.5 are set for *gamma, cost, and epsilon* parameters respectively. For XGBoost, default parameters are used.

4.2 Evaluation Procedure

Once predictions on the testing set are generated, the predicted outputs are compared to the real outputs using the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). Before training the models and predicting future quantities of cars sold each month, cross-validation is used to evaluate each model. K-fold cross-validation splits the training set into *k* subsets, it keeps *k-1* subsets for training and the last subset for validation, it fits the model and produces predictions using the model to be evaluated and uses an error scoring function to produce a score to compare different models. 5-fold cross-validation is used to evaluate SARIMA, SVR, and XGBoost with MAPE as the scoring function.

The RMSE is an evaluation metric that indicates how close the data is to the regression curve or predictions. It calculates the squared difference between the true and predicted values and finds the root of the mean of all squared errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_{i}-\hat{y}_{i})^{2}}{N}} \quad (8)$$

The MAE measures the absolute error between the true and predicted values and finds the mean of all absolute errors.

$$MAE = \frac{\sum_{i=1}^{N}|y_{i}-\hat{y}_{i}|}{N} \quad (9)$$

The MAPE value provides a measure of prediction accuracy for forecasting problems. MAPE measures the absolute percentage error between the true and predicted values by multiplying the difference between the values by 100 to get the percentage and then finding the mean of all absolute percentage errors.

$$MAPE = \left(\frac{1}{N}\right) * \sum_{i=1}^{N}\left|\left(\frac{y_{i}-\hat{y}_{i}}{y_{i}}\right) * 100\right| \quad (10)$$

Where:

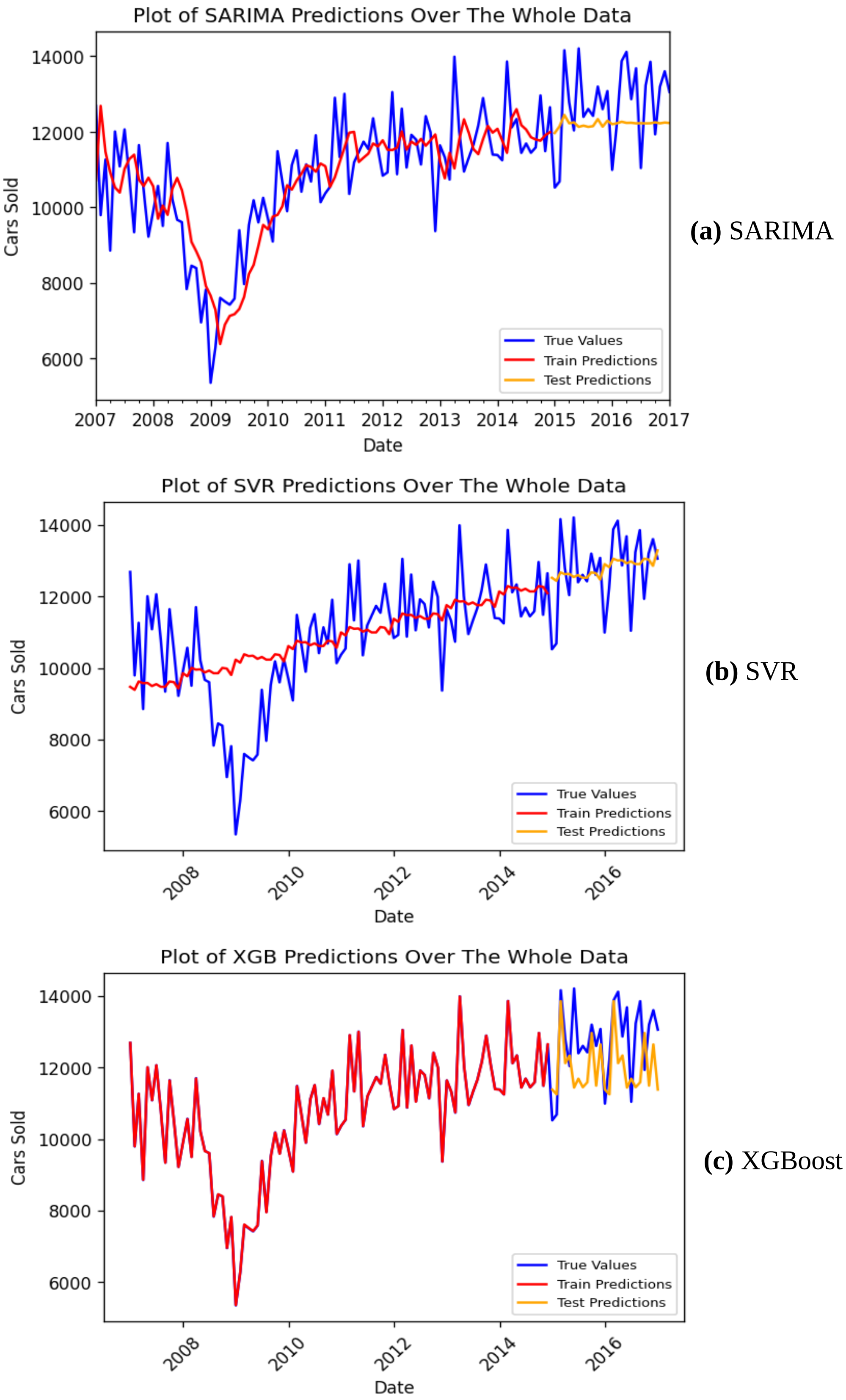
- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- N is the total number of predicted values

5. Results and Discussions

After the normalized predictions on both the training and testing sets are reversed to obtain the correct prediction values, the predictions on the testing set are compared with the true values and the RMSE, MAE, and MAPE values are compared for the three algorithms. The training and testing predictions are plotted on the whole dataset for visualization. Table 2 shows the RMSE, MAE, and MAPE values for each algorithm as well as the cross-validation score for 5-folds using MAPE as the scoring function. Figure 7 shows the plots of the predictions and true values for SARIMA, SVR, and XGBoost respectively.

Table 2. Error values for the three algorithms

	CV Score	***RMSE***	***MAE***	***MAPE***
SARIMA	3.37	1140.32	985.53	7.69
SVR	4.55	1026.01	802.16	6.54
XGBoost	5.74	1267.26	1045.54	8.06



Plot of SARIMA Predictions Over The Whole Data

(a) SARIMA

Plot of SVR Predictions Over The Whole Data

(b) SVR

Plot of XGB Predictions Over The Whole Data

(c) XGBoost

Figure 7. Predictions on the whole dataset for the three algorithms

From the results, SVR has the lowest RMSE, MAE, and MAPE error values than both SARIMA and XGBoost, this means that with default parameters SVR has the best performance on unseen data. However, this does not mean that SVR is the better choice to choose for this problem because when examining the model performance on the training sets in Figure 7, it is observed that SARIMA performs better than SVR on training data and captures the trend of the data, while SVR is increasing linearly and is not able to capture the pattern. Also, the SARIMA model has the lowest cross-validation score out of all the three algorithms, this indicates that SARIMA is the best algorithm to use for this problem, and with hyperparameters tuning, SARIMA can outperform SVR for this dataset. XGBoost performance on the training set is observed to be very accurate, however, it achieves the highest errors and cross-validation score on the testing set. The model performs too well on the training data but has the worst performance on the testing data out of the three algorithms. The training errors for XGBoost are calculated and compared to the testing errors to check for any presence of overfitting. From Table 3, the testing error for XGBoost is very high compared to the training error, which is very low, this indicates the presence of overfitting which explains why the model has the worst performance on unseen data.

Table 3. XGBoost error values on training and testing sets

	RMSE	***MAE***	***MAPE***
Training	6.99	5.86	0.06
Testing	1267.26	1045.54	8.06

In conclusion, with default parameters, the SVR model has the best performance on unseen data, while the XGBoost model leads to overfitting and achieves the worst performance on unseen data. The SARIMA model achieves the best performance on training data and can capture the trend, it also achieves the lowest cross-validation score, which indicates that SARIMA is the best model to train this dataset on, and with hyperparameters tuning it can outperform the SVR model.

6. References

[1] Kaggle, “New Car Sales in Norway”. Website, 2017. https://www.kaggle.com/datasets/dmi3kno/newcarsalesnorway

[2] T.Sharp, “An Introduction to Support Vector Regression,” *Towards Data Science.* Website, 2020. https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

[3] T. Chen, C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2016, pp. 785–794, doi:10.1145/2939672.2939785