A Comparison of Machine Learning Approaches for Predicting In-Car Display Production Quality

Luís Miguel Matos[1], André Domingues[1], Guilherme Moreira[2], Paulo Cortez[1], and André Pilastri[3]

[1] ALGORITMI Centre, Dep. Information Systems, University of Minho, Guimarães, Portugal
luis.matos@dsi.uminho.pt, a76953@alunos.uminho.pt, pcortez@dsi.uminho.pt

[2] Bosch Car Multimedia, Braga, Portugal
Guilherme.Moreira2@pt.bosch.com

[3] EPMQ, CCG ZGDV Institute, Guimarães, Portugal
andre.pilastri@ccg.pt

Abstract. In this paper, we explore eight Machine Learning (ML) approaches (binary and one-class) to predict the quality of in-car displays, measured using Black Uniformity (BU) tests. During production, the industrial manufacturer routinely executes intermediate assembly (screwing and gluing) and functional tests that can signal potential causes for abnormal display units. By using these intermediate tests as inputs, the ML model can be used to identify the unknown relationships between intermediate and BU tests, helping to detect failure causes. In particular, we compare two sets of input variables (A and B) with hundreds of intermediate quality measures related with assembly and functional tests. Using recently collected industrial data, regarding around 147 thousand in-car display records, we performed two evaluation procedures, using first a time ordered train-test split and then a more robust rolling windows. Overall, the best predictive results (92%) were obtained using the full set of inputs (B) and an Automated ML (AutoML) Stacked Ensemble (ASE). We further demonstrate the value of the selected ASE model, by selecting distinct decision threshold scenarios and by using a Sensitivity Analysis (SA) eXplainable Artificial Intelligence (XAI) method.

Keywords: Anomaly Detection · Automated Machine Learning · Deep Learning · Explainable artificial intelligence · Supervised Learning · One-class learning.

1 Introduction

The Industry 4.0 generates big data that can be used by Machine Learning (ML) algorithms to provide value [12]. In this work, we predict in-car display quality based on hundreds of assembly and functional tests from Bosch Car Multimedia. Currently, the manufacturer uses Black Uniformity (BU) tests to measure the display of solid black over the entire device screen. By adopting predictive ML models, the goal is to model the currently unknown relationships between the assembly and functional tests (the inputs) with the in-car display final quality (measured using BU tests), allowing to identify failure causes (e.g., screwing defect).

Most related works using ML to predict production faults consider a binary classification approach [1]. A less adopted approach is to use a one-class learning, such as

Isolation Forest (iForest) and deep dense Autoencoder (AE), which only uses normal examples during the training phase [9, 11]. Moreover, the best ML algorithm is often selected by using trial-and-error experiments that consume time. The Automated ML (AutoML) and Automated Deep Learning (ADL) concepts were proposed to reduce the ML analyst effort [4]. Within our knowledge, there are no studies that use AutoML or ADL to predict production failures. In addition, there are no studies that model BU quality using ML based on assembly and functional tests. Thus, the novelty of this paper comes from a practical application point of view, where we compare several ML algorithms (e.g., one-class, AutoML and ADL) and estimate their potential value for the analyzed industrial use case. In particular, we explore eight ML methods: one-class - iForest and deep AE; and binary classification - Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), Deep Feedforward Neural Network (DFFN), AutoML Stacked Ensemble (ASE) and an ADL. To evaluate the methods, we collected a dataset with around 147 thousand in-car display quality tests and adopt two evaluation schemes: an initial time ordered holdout train and test split and then a more robust rolling windows, which simulates several training and test iterations through time. Finally, we demonstrate how the best predictive ML model can provide a value for the analyzed industrial domain by selecting diverse decision thresholds and by using a Sensitivity Analysis (SA) eXplainable Artificial Intelligence (XAI) method [2].

2 Materials and Methods

2.1 Industrial Dataset

For this study, we collected 146,536 records related with in-car displays produced from a Bosch Car Multimedia in the year of 2020. The left of Fig. 1 exemplifies a produced in-car display. For the analyzed product, there are three main intermediate quality tests that are executed during the production process: gluing and screwing process (both executed during assembly), and functional tests (performed after assembly and prior to BU testing). The final quality is assessed by using a BU test that returns a percentage (the higher the value, the better is the display screen). The right of Fig. 1 shows the result of a failed BU test (due to a large detected red region).

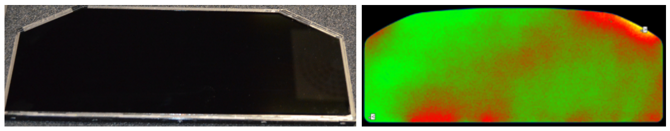

Fig. 1: Example of a produced display (left) and a BU failure test (right).

In an initial phase of the ML project, the manufacturer provided the raw BU output values and an initial set of input features (a subset of the screwing process tests) that were considered by the manufacturer as more relevant to influence the BU quality, resulting in dataset A. Since the obtained ML results were not considered sufficiently good (as shown in Section 3), in a second ML project stage, the full intermediate quality tests (including the gluing process and functional tests) were also requested, leading to the creation of dataset B. All inputs are numeric the Carrier Cavity (CC) and System (SCC) attributes, which are categorical with 8 and 7 distinct levels (used in both A and B sets). In total, dataset A includes 110 screwing input variables (108 numeric and 2 categorical), while dataset B has 1032 input assembly and functional attributes (1030 numeric and 2 categorical).

The data preprocessing involved the transformation of the BU values into a binary target, by using the manufacturer quality rule: $y =$“Fail” if BU<40% else $y =$“Pass”. The output target (y) is unbalanced, including only 2,138 abnormal instances. When the ML algorithm requires a numeric output (e.g., LR, DFFN), we assume the “Fail” class as the positive concept, thus transformed into $y = 1$ (if true) or $y = 0$ (if false). Moreover, the CC and SCC categorical variables were encoded into binary numeric ones (within $\{0,1\}$) by adopting the popular one-hot encoding. Finally, the numeric variables were rescaled into the [0,1] range by adopting the popular min-max normalization.

2.2 Anomaly Detection Methods

All ML methods were implemented by using the Python language and the following modules: `scikit-learn` – for iForest, DT, LR and RF (https://scikit-learn.org/stable/); `TensorFlow` – for AE and DFFN (https://www.tensorflow.org/); and `H2O` – ASE and ADL (https://docs.h2o.ai/).

The iForest is a recently proposed one-class ML algorithm [7]. The `scikit-learn` iForest implementation provides a decision score that ranges from $\hat{y}_i =$-1 (highest abnormal score) to $\hat{y}_i = 1$ (highest normal score). In order to obtain an anomaly probability score ($d_i \in [0,1]$, for an input example i), we rescale the iForest scores by computing $d_i = (1-\hat{y}_i)/2$.

Autoencoders (AE) compresses and encode data into a lower-dimensional representation by assuming a bottleneck layer (with L_b hidden units) [5]. Let $(L_I, L_1, ..., L_H, L_O)$ denote the structure of a dense (fully connected) DFFN architecture with the layer node sizes, where L_I and L_O represent the input and output layer sizes and H is the number of hidden layers. The proposed AE assumes $L_I = L_O$, a symmetrical encoder and decoder structure (e.g., $L_1 = L_{O-1}$) and the popular ReLu activation function is used by all neural units. In the encoder component, the number of hidden layer units decreases by half in each subsequent hidden layer until the bottleneck size (L_b) is reached: $L_1 = L_I/2$, $L_2 = L_1/2$, and so on. Each hidden layer is also attached with a Batch Normalization layer. When adapted to anomaly detection, the AE training algorithm is only fed with standard (normal) instances, aiming to generate output values that are identical to its inputs. In this work, the AE is trained with the Adam optimizer using a batch size of 1024, 100 epochs and early stopping (using 10% of the training data as the validation set). The Mean Absolute Error (MAE) is used as the loss function and reconstruction error: $MAE_i = \sum_{k=1}^{n} \frac{|x_{i,k} - \hat{x}_{i,k}|}{n}$, where $x_{i,k}$ and $\hat{x}_{i,k}$ denote the AE input and output value for the i-th data instance and k-th input or output node. The reconstruction MAE error is used as the decision score $d_i = MAE_i$, where higher reconstruction errors should correspond to a higher anomaly probability. In preliminary experiments, using the training data from the first partition (P1, Section 2.3), a grid search was used to search for the best $L_b \in \{2,4,8,16\}$ value that provided the lowest reconstruction error. The best result was achieved using $L_b = 8$, which was kept fixed in the remaining AE experiments.

Focusing on the supervised learning models used, the LR, DT and RF algorithms were set to output an anomaly class probability (d_i for instance i). Turning to the more complex deep learning DFFN model, it assumes the base dense architecture presented in [8] with no additional tuning. We note that this DFFN is adopted as a default supervised deep learning model since the ADL (described below) already performs a substantial DFFN hyperparameter selection. The default DFFN has H=9 hidden layers, under the structure (L_I, 1024, 512, 256, 128, 64, 32, 16, 8, 2, 1). The ReLu activation function is used in the hidden layers, while the logistic function is adopted in the output node, in order to output an anomaly class probability (d_i). To avoid overfitting, we network includes a Dropout with the values 0.5 and 0.2 in the fourth and sixth hidden layers. The DFFN assumes the same AE Adam training, with the difference that a different loss function is used (binary cross entropy).

The ASE and ADL methods were implemented by using the AutoML H2O tool. The ASE first trains 5 distinct regression algorithms: RF, Generalized Linear Model (GLM), XGBoost, Gradient Boosting Machine and a default DFFN network. Then, it employs a Stacking Ensemble (SE), which uses all previously trained models to generate inputs for another GLM model. As for ADL, it uses a dense DFFN, with the H2O automatically tuning 7 of its hyperparameters (e.g., number of hidden units per layer, learning rate). During the AutoML and ADL search, the H2O tool was configured to maximize the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) analysis [3], using a 5-fold cross-validation. Preliminary experiments using the training data from the first partition (P1, Section 2.3), allowed to set the stopping criterion for AutoML and ADL, which was fixed to stop after a time limit of 300 (dataset A) and 1,200 (dataset B) seconds.

Descriptive knowledge can be directly extracted from any trained ML model (e.g., ensemble, deep learning) by applying a SA XAI method. In this paper, we adopt the computationally efficient one-dimensional SA (1D-SA) [2], which holds all ML inputs at their average values, except one target input, which is changed with L=7 distinct levels. The ML output responses are stored, allowing to compute an input relevance, which is proportional to the Average Absolute Deviation (AAD) measure applied to the responses, and Variable Effect Characteristic (VEC) curves, which plot the SA ML responses for each input. This SA XAI was implemented by using the `rminer` package of the R tool (https://CRAN.R-project.org/package=rminer) with the following parameters: SA `method=1D-SA`; default values for other parameters (e.g., number of levels `L=7, measure = "AAD"`).

2.3 Evaluation

We adopt two evaluation schemes. Assuming time ordered records, the data (146,536 instances) is divided into two main partitions (Fig. 2): **P1** - with the oldest 70% elements (102,575 records); and **P2** - with the more recent 40% examples (58,613 records). The first partition (P1) was used to explore an initial single train and test Holdout Split (HS). The second partition (P2) was used to execute a more robust Rolling Window

(RW) procedure [13], which simulates a real classifier usage through time by adopting a fixed training window (W), which is rolled in different iterations (with step size of S), generating U training and testing updates. As shown in Fig. 2, the test data from the P1 (HS) and P2 (RW) partitions do not overlap. The HS uses the oldest 70% P1 data to fit a model (71,802 instances, includes model selection and training) and the remaining 30% (30773 records) to test the predictive capability of the ML methods. The first partition test results for dataset A (Section 3) were shown to the industrial experts, which identified a higher predictive performance need. This triggered the collection of further intermediate quality inspection tests (e.g., functional tests), that resulted in dataset B. Then, dataset B was also evaluated on P1 test data, obtaining improved results. The second RW evaluation scheme was applied to the best dataset B ML algorithms. In the first RW iteration ($u = 1$), the more recent W examples from P1 were used as the ML fit data, allowing to predict the next test T examples. In the second iteration ($u = 2$), the training data is updated with S newer examples allowing to fit a new ML model and perform T predictions, and so on. In total, the RW results in $U = \frac{P2-(W+T)}{S}$ model updates, where $P2$ is the second data partition length. We use $U = 20$ RW iterations by fixing the values: W =25,000, T =5,000 and $S = 1,400$.

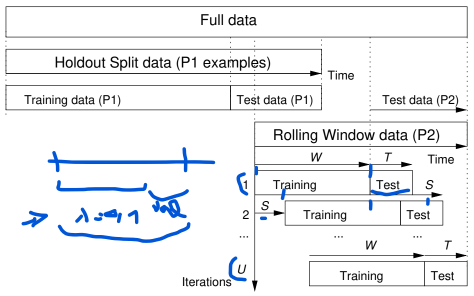

Fig. 2: Schematic of Holdout Split (HS) and Rolling Window (RW) evaluations.

The predictive classification performance is based on the ROC curve [3]. When a classifier outputs a decision score d_i, the class can be interpreted as positive if $d_i > K$, where K is a fixed decision threshold, otherwise it is considered negative. With the class predictions there will be True Positives (TP), True Negatives (TP), False Positives (FP) and False Negatives (FN). The ROC curve shows the performance of a two class classifier across all $K \in [0,1]$ values, plotting one minus the specificity (x-axis), or False Positive Rate (FPR), versus the sensitivity (y-axis), or True Positive Rate (TPR). The discrimination performance is given by the $AUC = \int_0^1 ROCdK$. The AUC metric has two main advantages [10]. Firstly, when the data is unbalanced (which is our case), the interpretation of the goodness of the metric values does not change. Secondly, the AUC values can be interpreted as: 50% - performance of a random classifier; 60% - reasonable; 70% - good; 80% - very good; 90% - excellent; and 100% - perfect. If needed, the best K threshold can be selected by using the ROC curve of a validation set (Section 3). Since the RW produces several test sets, for each RW iteration we store the AUC value on test data. We also record the computational effort, in terms of the total training time (in s) and prediction response time for one instance (in μs) when using an 2.4 GHz i9 Intel processor. To aggregate all $u \in \{1,...,U\}$ execution results, we compute the median values, it is less sensitive to outliers when compared with the average. The Wilcoxon non parametric test is used to check if paired differences are significant [6].

3 Results

Table 1 presents HS predictive test results. For all compared ML algorithms, the usage of the first dataset (A) results in a lower class discrimination capability, with the AUC values ranging from 51% (almost random classifier for DFFN) to 69% (LR). When using more inputs (dataset B), there is a substantial improvement in the BU anomaly detection, with all ML algorithms presenting a much higher AUC value.

Table 1: Anomaly detection results (AUC values) for P1 and the HS evaluation (values higher than 0.75 are in **bold**).

	Unsupervised		**Supervised**					
	iForest	AE	DT	LR	RF	DFFN	ASE	ADL
Dataset A	0.61	0.60	0.55	0.69	0.64	0.51	0.52	0.64
Dataset B	0.71	**0.76**	0.60	**0.87**	**0.84**	**0.87**	**0.84**	0.60

The ML algorithms that obtained an AUC>75% using Dataset B in Table 1 were selected for the second stage evaluation: AE, LR, RF, DFFN and ASE. The respective RW results are shown in Table 2 and left of Fig. 3. For comparing purposes, we also present the dataset A RW results in Table 2. Similarly to the previous HS evaluation, the best RW results were achieved when using dataset B. In effect, for all tested ML, the AUC differences between dataset B and A are statistically significant (p-value$<$ 0.05). Using dataset B, the best overall anomaly detection performance was obtained by ASE (AUC of 92%), followed by RF (91%), LR and DFFN (89%) and AE (87%). The individual AUC results for each RW iteration are plotted in the left of Fig. 3. Both ASE and RF (purple and green curves) present a consistent excellent discrimination ($\geq$90%) after iteration $u = 6$. Regarding the computational effort, LR and RF provide the fastest training and predict response times. Yet, even the more demanding ML algorithm (ASE) requires a computational effort that is acceptable for the analyzed domain. Thus, we select the ASE model for the remainder demonstration results.

To demonstrate the ASE value, we adopt the last RW iteration ($u = 20$) results, using the most recent non overlapping RW test set ($u = 17$) as a validation set for selecting a K threshold. The ROC curve is shown in the right of Fig. 3, allowing to define several FPR and TPR trade-offs. Five thresholds (K) were fixed and applied to the last RW iteration test results (Table 3), confirming that the selected thresholds (from $u = 17$) correlate highly with similar sensitive and specific test (from $u = 20$) trade-offs results.

Table 2: Anomaly detection results for P2 and the RW evaluation (best AUC values per dataset in **bold**; best global AUC is <u>underlined</u>).

	AUC					**Training** (s)					**Predict** (μs)				
	AE	LR	RF	DFFN	ASE	AE	LR	RF	DFFN	ASE	AE	LR	RF	DFFN	ASE
Dataset A	0.66	0.72	0.66	0.71	0.66	48.6	0.4	1.3	8.1	300.0	30	1	10	30	50
Dataset B	**0.87**	**0.89**	**0.91**	**0.89**	<u>**0.92**</u>	88.9	3.6	2.0	13.7	1200.0	150	4	10	100	60

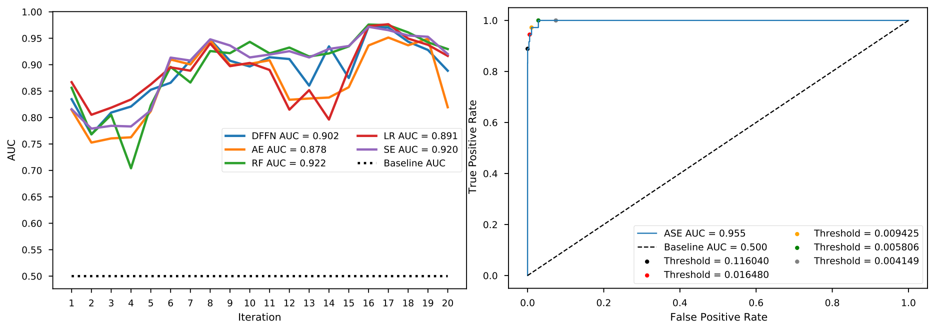

Fig. 3: Evolution of the AUC measure for the distinct RW iterations (left) and the ROC curve for ASE and iteration u=17 (right).

Table 3: Class label ASE prediction results for five threshold values ($u = 20$).

K	**TP**	**TN**	**FP**	**FN**	TPR	FPR
0.004149	56	4035	239	41	90.3%	18.3%
0.005806	49	4534	403	13	79.0%	8.2%
0.009425	41	4698	239	21	66.1%	4.8%
0.016480	30	4803	134	32	48.4%	2.7%
0.116040	13	4910	27	49	30.0%	0.5%

Secondly, we applied the SA XAI approach to the ASE model that was fit using the last RW training data ($u = 20$). The left of Fig. 4 plots the top 20 relevant input variables from dataset B (total of 1032 inputs). For instance, the most influential input is related with a functional test (total relevance of 6%). The top 20 inputs account for 51% of the influence in the ASE model. The right of Fig. 4 shows the VEC curves for the top 5 input variables. The plot clearly reveals that the most influential input (Func02_T4105.02) produces the largest ASE output response change, where an increase in the numeric test value results in a nonlinear BU failure probability decrease.

4 Conclusions

In this paper, we use ML to model BU quality tests of in-car displays based on assembly and functional intermediate tests. A large set of comparative experiments was

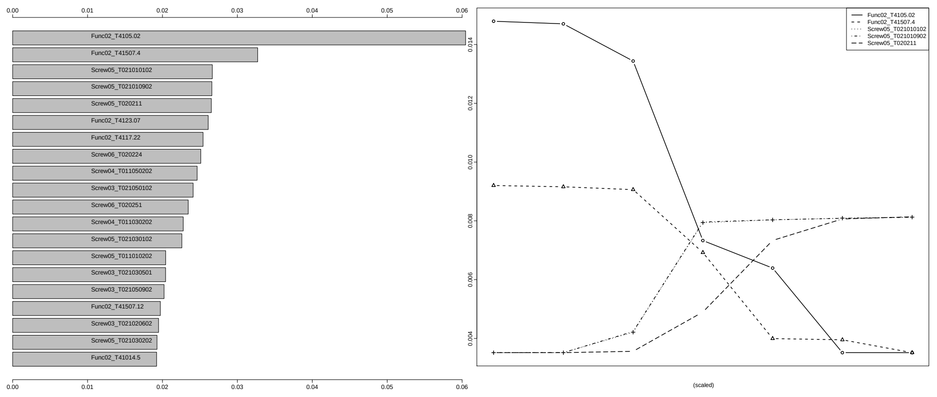

Fig. 4: Extracted knowledge from the ASE model ($u = 20$) using the SA XAI method: top 20 relevant input variables (left) and top 5 VEC curves (right).

held, involving 147 thousand in-car display records. Overall, an excellent discrimination level was obtained by the ASE model (92%) when using both assembly and functional inputs (input set B), followed by the RF (91%), LR and DFFN (89%) and AE (87%). The AE results are particularly appealing for initial product production stages, since the model does not require labeled data and anomaly instances are rare. As for the best predictive model (ASE), it requires a computational effort that is affordable in this domain. The ASE model is particularly appealing for the analyzed industrial domain because it can be automatically adapted to new dynamic data changes, without needing an ML expert to tune or select the model. Moreover, we have shown that more specific or sensitive decision thresholds can be selected by using a validation set, producing a similar classification performance on unseen data. Furthermore, by using a SA XAI procedure, we have shown how descriptive knowledge can be extracted from a trained ASE model, which is valuable for BU failure cause identification. The obtained results were discussed with the manufacturer experts, which returned a very positive feedback.

In future work, we aim to deploy the proposed ASE in a real industrial setting. In addition, we plan to define intermediate production checkpoints before reaching the final BU test. This would allow the setting of quality assessment ML models at early production stages, thus saving production time and costs, and ultimately reducing the number of BU faulty tests.

Acknowledgments

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 39479; Funding Reference: POCI-01-0247-FEDER-39479].

References

1. Angelos Angelopoulos, Emmanouel T. Michailidis, Nikolaos Nomikos, Panagiotis T., Antonis Hatziefremidis, Stamatis V., and Theodore Z. Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects. *Sensors*, 20(1), 2020.
2. Paulo Cortez and Mark J. Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Inf. Sci.*, 225:1–17, 2013.
3. T. Fawcett. An introduction to ROC analysis. *Patt. Recognition Letters*, 27:861–874, 2006.
4. Luís Ferreira, P.M. Pires A. Pilastri, C.M. Martins, and Paulo Cortez. A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost. In *International Joint Conference on Neural Networks, IJCNN 2021*. IEEE, 2021.
5. G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. cited By 9376.
6. Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.
7. Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proc. of the 8th IEEE Int. Conf. on Data Mining (ICDM), Pisa, Italy*, pages 413–422. IEEE, 2008.
8. Luís Miguel Matos, Paulo Cortez, Rui Mendes, and Antoine Moreau. Using deep learning for mobile marketing user conversion prediction. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019.
9. D. Pandya, A. Srivastava, A. Doherty, S. Sundareshwar, C. Needham, A. Chaudry, and S. KrishnaIyer. Increasing Production Efficiency via Compressor Failure Predictive Analytics Using Machine Learning. In *Offshore Technology Conference, OTC-28990-MS*, 2018.
10. Pedro José Pereira, Paulo Cortez, and Rui Mendes. Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction. *Expert Systems with Applications*, 168:114287, 2021.
11. Diogo Ribeiro, Luíis Miguel Matos, Paulo Cortez, Guilherme Moreira, and André Pilastri. A Comparison of Anomaly Detection Methods for Industrial Screw Tightening. In *21th International Conference on Computational Science and its Applications (ICCSA)*. Springer, September 2021.
12. António João Silva, Paulo Cortez, Carlos Pereira, and André Pilastri. Business analytics in Industry 4.0: A systematic review. *Expert Syst. J. Knowl. Eng.*, 2021.
13. L.J. Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Forecasting Journal*, 16(4):437–450, 2000.