International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P), Volume-11, Issue-10, October 2021

LSTM and LSTM Autoencoder for Predictive Maintenance Approach Applied to Slagging Formation

Ulrich BIAOU, Donald MOUAFO, Michael BOCQUET

Abstract: The industrial system relies on the manufacturing infrastructure which is susceptible to failures. These failures often induce severe consequences such as downtime and expensive corrective maintenance actions. An optimal maintenance management system is therefore essential for anticipating failures, and likely to provide cheaper and better service quality. In this paper, we propose a data-driven approach for predicting the slagging formation leading to downtime of the production system and expensive maintenance actions in coal-power plant industry. The approach relies on Long Short-Term Memory (LSTM) Neural Networks (NNs) for Remaining Useful Life (RUL) prediction by either forecasting or classification methods. We suggest two stages stacked model architecture consisting of an unsupervised LSTM autoencoder for feature extraction from high correlated multivariate time series data and LSTM NN for prediction. In addition, we developed an alternative approach relying on an optimized LSTM model serving as benchmark for model evaluation. The obtained results show that the stacked model outperforms an optimized simple LSTM model for RUL forecasting. For classification both models lead to high performances such that the discrimination could mitigated as the simple LSTM is gingerly optimized, the stacked model remains preferable for stability reasons.

Index Terms: LSTM, RUL, slagging, predictive maintenance.

I. Introduction

In the new generation of industries refer to as industry 4.0, the production infrastructures are connected to the digital environment thanks to the internet of things (IoT) technologies (Zonta et al., 2020). This enables collecting big amounts of data containing crucial information from manufacturing process and system dynamics. An appropriate processing of such data can likely permit accessing valuable insight and provides key input to the prognostic health management (PHM) critical for maintenance cost reduction. This gives promise to data driven predictive maintenance (PdM) that is a maintenance management strategy consisting of scheduling maintenance actions at the appropriate time before equipment failure.

Three approaches of implementing PdM exist in the literature [2]; (Zonta et al., 2020). The Physical model-based approach relying on mathematical modelling of the equipment's conditions. The knowledge-based approach which is a hybrid method combining statistics, domain expertise and other fuzzy logic to develop failure prediction. The Data driven approach based on machine learning (ML) models, increasingly adopted the last recent years as the most promising strategy for PdM solutions. Based on the available information on historical data, ML models for PdM can be either supervised or unsupervised [3]. In the case of not clear knowledge of historical failures, anomaly detection methods can be applied as there is no requirement of labeled data. Supervised learning in contrast requires labelled data for model training and testing. This implies the availability of sufficient information on historical faults in other to conduct the labelling. Depending on the situation, the suitable PdM solution can either be a regression [1] or a classification [4]. The purpose of regression is to predict the remaining useful life (RUL) while in classification, the concern can be either binary or multi class. Binary classification often concerns predicting the likelihood of failure to occurs within a given future time period. It happens to target different future time periods at once or instead, different type of futures using multiclass classification [5].

ML models for PdM have been constantly increasing these recent years in a variety of industrial sectors [2] and [1]. For instance, in automotive industry, Random Forest (RF) has been employed to develop a predictive model for air compressor failures based on available warranty and logged vehicle data [6]. A similar model was employed for faults detection of stator winding short circuit in squirrel-cage induction motors [7]. Also, Fault prediction of gearbox systems with SVM model using sound and vibration signals has been reported [8]. In semiconductor microelectronic manufacturing, [9] reported on an ensemble technique consisting of Generalized Linear Model, RF, GBoosting, and ANNs for the prediction of RUL of equipment components using sensor data. Similarly, in the retail industry [10] reported on the fault prediction on refrigeration and cold storage system of supermarkets using historical data of temperature based on RF models. PdM has also proven important promise in the Oil & gas industry where ML based IA approaches are widely used for maintenance management of production health equipment [11]. More specifically, [12] proposed a neural network model to predict corrosion induced leakage in petrochemical pipelines using sensor data of pressure, temperature and instantaneous liquid flow rate. They addressed the problem as a multi-class classification with classes ranging from the healthiest state to failure and found that LSTM performs better than RF.

Data driven failure detection approach can apply to any industrial sector provided that historical data with suitable

Dr. Ulrich BIAOU, CEO BESTTIC, Famars, France.

Dr. Donald MOUAFO, BESTTIC, Famars, France.

Dr. Michael BOCQUET, Associate Professor, OAE Department, IEMN UMR CNRS 8520. Université de Valenciennes et du Hainaut Cambrésis. 59313 Valenciennes cedex 9.

properties are available. In this paper, we develop a model for slagging prediction. Slagging is a phenomenon that appears in boilers of coal-fired power plants and induces large and expensive maintenance actions as well as reducing the lifetime of the production devices. An efficient slagging prediction method can enable better anticipation of maintenance actions in order to reduce the downtime of boilers and the cost of the maintenance. In addition, anticipating the slagging event also enables reducing the volume of pollution gases participating thus in minimizing the environmental impact of electrical production. In the literature, most of the solutions for slagging prediction rely on physical model-based approaches exploiting the thermo-chemical conditions within the boiler [13] [14]. In some cases, these methods enable to calculate the slagging index which guides maintenance management [14]. However, they require detailed knowledge of the physical nature of the thermo-chemical mechanisms leading to the slagging formation. Unfortunately, these mechanisms are known to be extremely complex making their theoretical formulation complicated. As consequence, model-based approaches for slagging prediction have limited efficiency and do not generalize well. We propose an approach for slagging prediction based on LSTM and LSTM autoencoder using the sensor data of boilers in coal-fired power plant. The rest of the paper is organized as follows: In Section 2, we describe the data set and present the maintenance request. In Section 3, we present the methodology and provide a general overview of unsupervised LSTM autoencoder and LSTM modules together with the details of the model architecture. In Section 4, we present model evaluation followed by the discussion. The paper ends with the conclusion.

II. DATA SOURCE AND MAINTENANCE REQUEST

A. Data Source

The data source consists of multivariate time series sensor data collected on the boilers of a coal-fired power plant. The data set includes three of non-necessarily consecutive years data record spread in 3 files (each of one year data record). Each file consists of 116 numerical features recorded every 5 min time step. The features include data such as flue gas temperature and pressure, the flow rate or active electrical power or order indicator of power plant. The Table 1 summarizes the available data file and the corresponding size and records.

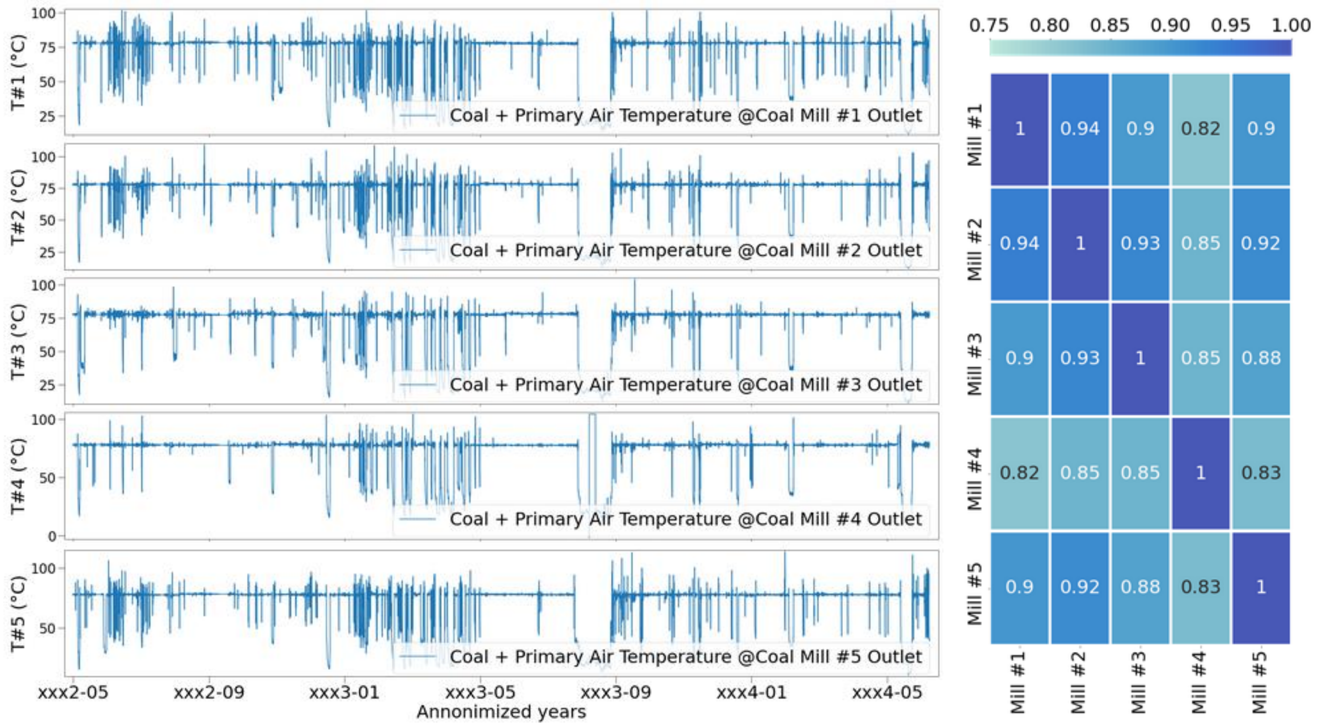
The data are provided by “Energies de Portugal” (EDP) company and available on EDP data repository. They were collected by the sensors deployed on different equipments of the production chain. Often, many sensors are replicated at different positions of the same equipment to ensure complete information. The number of replicates depends on the equipment and the sensed information. For example, to probe the temperature of the drum, four sensors are deployed at four different parts of the drum while only one sensor is dedicated to the pressure within the drum. This is because the inhomogeneity of the temperature is more likely than that of the pressure.

File Name	Shape
boiler_unitx_xxx0.csv	(55284, 116)
boiler_unitx_xxx3.csv	(105408, 116)
boiler_unitx_xxx4.csv	(105408, 116)

Table 1. Summary of data files recorded on one boiler denoted ‘boiler X’

B. Data Exploration And Cleaning

The Fig. 1. presents 5 features recording the temperatures within the coal mills equipment. The Fig. 1. (left panel) displays the temporal behavior of the 5 temperatures. While the right panel presents the heatmap of the Pearson correlation coefficients between the five temperatures. One notices a strong correlation between these five sensors temperatures. This is a general tendency not only for the features collected from sensors replicated on the same equipment, but also other features of the data set and suggests deeper attention to the process of feature extraction and selection. It is worth noting that the analysis discussed in this section goes beyond the features highlighted in Fig. 1 & 2 and concerns all other features of the data set. Classical methods commonly used to concomitantly handle redundancy in the data set and reduce the dimensionality such as principal component analysis (PCA) failed to properly extract representative features, probably due to temporal correlation. Feature extraction has required more advanced methods based on LSTM autoencoder as will be detailed later. Alternatively, a relevant approach for dimensionality reduction precisely consisted in selecting one feature from each group of replicated features. This selection process results to 47 features used to develop the benchmark LSTM model. The data set contains missing values which are not always randomly distributed. Instead, they are sometimes expanded on several consecutive days.



	Mill #1	Mill #2	Mill #3	Mill #4	Mill #5
Mill #1	1	0.94	0.9	0.82	0.9
Mill #2	0.94	1	0.93	0.85	0.92
Mill #3	0.9	0.93	1	0.85	0.88
Mill #4	0.82	0.85	0.85	1	0.83
Mill #5	0.9	0.92	0.88	0.83	1

Figure 1: Distribution and correlation of historical temperatures of the output mixture of coal Mill. The left panel presents the line plot, the legend corresponds to the exact names of the features. The right panel presents the heatmap of correlation coefficient and the sticks label represent the truncated names of the corresponding features.

Decomposition analysis did not demonstrate any noticeable seasonality. In addition, autocorrelation analysis did not lead a positive conclusion. We filled missing values using rolling mean methods within 30 min time windows. The Fig. 2 (a, c) illustrates the distribution of 2 random features and Fig 2 (c & d) presents their corresponding box plot and statistical distribution. The box plots demonstrate an abundant presence of outliers which are also very spread. The statistical distribution of both features shows bimodal behavior which the smaller reveals the distribution of outliers. To ensure homogenous contribution of all numerical features' values during model training, we applied
Min-Max normalization. Furthermore, in order to minimize the intrinsic noise, the data were resampled by taking the mean every 30 min windows corresponding to every 6 instances.

C. Maintenance Request

The structure of the dataset induces important constraints in their preparation prior to train the model for slagging prediction. Indeed, for the entire period, only one slagging date is known. In general, for PdM, the data must be organized in cycles of operations. In case of classification, each instance of the cycle must be properly labelled. Ideally, each cycle consists of historical operations ending with failure after which a new cycle starts. Thus, no matter how the data is organized, one must be able to identify the beginning and end of each cycle. As described in section 2, the dataset provided by EDP consists of three years continuous acquisition data with a single known slagging date. This makes the repartition of data in cycles unrealistic for the entire period. Therefore, in this paper, we limited the study to the time interval not exceeding 60 days before the slagging date as there was no other slagging event reported during this period. We adopted both regression and classification approaches. For regression, starting 45 days before the known slagging day, the RUL is defined as the remaining number of days until the slagging following the Eq. 1. Therefore, all the data instances of a given date share the same RUL values.

$$RUL_{i^{th}} = slag_{date} - i^{th}date \tag{1}$$

Where $slag_{date}$ refers to the slagging date and $i^{th}date$ the date of the i^{th} data instance. For the classification, we adopted a binary approach and, instead of labeling the exact slagging date as failure and the other as normal, we implemented an anticipation logic. In this context, we define a failure date as $failure_{date} = slag_{date} - 15\ days$. Hence, the i^{th} data instance is labeled "1" if the corresponding i^{th} date is equal to the failure date, otherwise it is assigned the label "0". To guarantee the predictive aspect of the approach, the historical data starts 60 days before and ends at the failure day. This provides 15 days anticipation for any true positive prediction.

$$Label_{i^{th}} = \begin{cases} 0, & if\ i^{th}date\ =\ Failure_{date} \\ 1, & if\ i^{th}date\ \le\ Failure_{date} \end{cases} \quad (2)$$

With $Label_{i^{th}}$ is the label of the i^{th} data instance. This labeling method results in high class imbalance which requires attention when training the model.

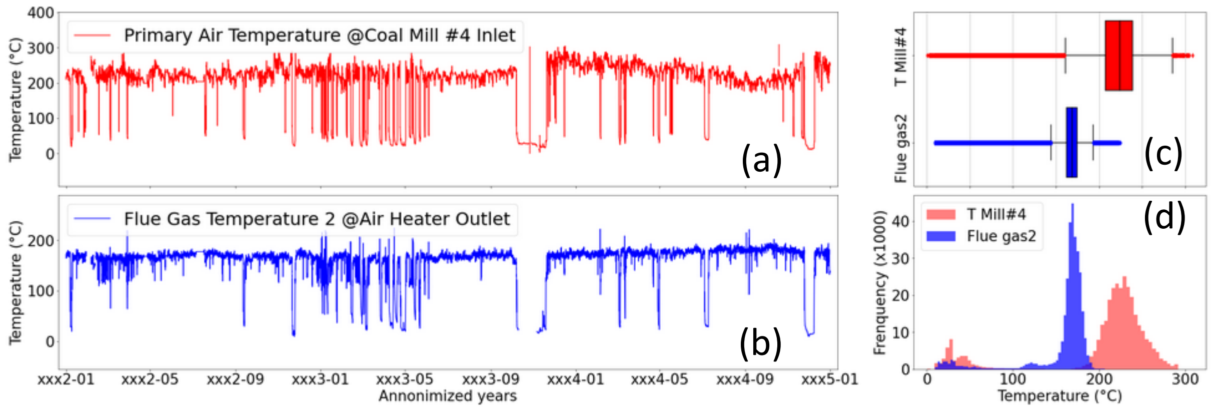


Figure 2: Distribution of two random features. (a) and (b) present the line plots of the two features. (c) presents the box plot of the two features. (d) presents the histogram plot of the two features.

III. METHODOLOGY

Addressing the maintenance requests presented in section 2.3 requires considering two main characteristics of the data. First, the high dimensionality with important features correlation highlighting information redundancy. Second, the temporal correlation which integrates the hidden pattern in historical data resulting in the slagging. To address the two challenges, we propose a stacked model consisting of an unsupervised LSTM autoencoder NN [15] for feature extraction and LSTM NN [16] for prediction. Indeed, both LSTM autoencoder and LSTM NN have proven stat-of-art performance respectively for feature extraction from multivariate time series data while keeping the maximum information from the hidden pattern of temporally correlated data [16]; [17].

A. LSTM Autoencoder

An autoencoder is an unsupervised NN that aims to learn the best representation of the input data. Its architecture generally consists of an input layer, an output layer together with hidden layers consisting of two symmetric neural networks representing the encoder and the decoder both of which sandwich a latent space representation layer of the input dataset. When an input data is fed to the auto-encoder, it is compressed by the encoder into the latent space, whereas the decoder decompresses the encoded representation into the output layer. The encoded-decoded output is then compared with the initial input data and the error is back propagated through the architecture to update the weights of the network [16]. In particular, when feeding the autoencoder with an input data $X = \{x^1, x^2, \ldots, x^m\}$ with m the number of features ($X \in \mathbb{R}^m$), the encoder compresses X into an encoded representation $Z = e(X) = \{z^1, z^2, \ldots, z^n\}$ ($Z \in \mathbb{R}^n$) with $n < m$. The decoder then reconstructs the encoded representation back to a $\mathbb{R}^m$ representation $\hat{X} = d(Z) = \{\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^m\}$ that is as close as possible to X. This training is conducted by minimizing a reconstruction loss function such as Mean Square Error in this case (MSE= $\sum_x \|x - \hat{x}\|^2$). Doing so, the autoencoder is forced to reduce the data dimension while keeping the major information of internal data structure. Therefore, it is a powerful tool for feature extraction. Depending on the type of NN used for encoder and decoder, different variant of autoencoder have been proposed in the literature including Vanilla autoencoder, convolutional autoencoder, regularized autoencoder, and LSTM autoencoder. The latter has demonstrated improved potentiality in encoding multivariate time series data. [16] exploited the automatic feature learning capabilities of LSTM autoencoder combined with LSTM NNs to demonstrate an end-to-end model for forecasting rare events in drive demand. More recently, [17] demonstrated a stacked LSTM autoencoder/LSTM based model outperforming simple LSTM model for anomaly detection in the supply chain management. This demonstrates the capability of LSTM autoencoders for capturing hidden nonlinear correlation in temporal data when conducting encoded representation of input data.

B. Long Short-Term Memory Networks

LSTM is a particular type of RNN that consists of a chain of repeated NN modules allowing at a given time to retain long-term memories from many timesteps back. In standard RNNs, the repeated module has a simple structure with single tanh layer making the memory persistence not efficient. This problem is accurately handled in LSTM architecture with a repeating module composed of three gates and tanh layers. Each gate including the forget gate, the input gate, and the output gate consists of a sigmoid layer and a pointwise multiplication operation [17] [18]. A key quantity to LSTM, namely cell state C_t, runs straight down the entire chain of modules. In each module, it undergoes minor linear interactions which enable removing or adding information decided respectively by the forget gate and input gate based corresponding sequence data x_t and the output h_{t-1} of the previous modules [17].

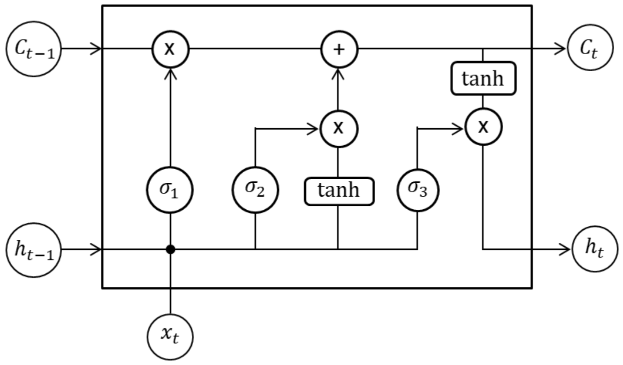


Figure 3: Internal structure of an LSTM neural network module.

When new input at time t, denoted $X[t] = X_t$ ($X_t \in \mathbb{R}^m$) is fed to an LSTM module, it is first concatenated with the output h_{t-1} from the previous time step $t-1$. Then, the forget gate decides what old information should be forgotten by computing a number $f_t \in [0, 1]$ following Eq. (3). The value 0 means "let nothing through" while the value 1 means "let everything through".

$$f_t = \sigma_1(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

Where, W_f is the weight matrices and b_f the bias of the forget gate. Following, the information to update the cell state is processed. The decision value i_t is determines in the input gate layer (Eq. (4) along with the vector of candidate values $\hat{c}_t$ (Eq. (5) computes by the tanh layer.

$$i_t = \sigma_2(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\hat{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (5)$$

From, the previously computed quantity, the new updated cell state C_t is subsequently computed (Eq. 6) (W_i, W_c) and (b_i , b_c) are respectively the weight matrices and the biases of the input gate and tanh layer.

$$C_t = f_t * C_{t-1} + i_t * \hat{c}_t \quad (6)$$

Finally, the module's output h_t is computed by applying the output gate to the candidates generated by the last tanh layer Eq. (7) and Eq. (8).

$$O_t = \sigma_3(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = O_t * tanh(C_t) \quad (8)$$

Where W_o and b_o the weight matrix and the bias of the output gate.

C. Model Architecture

The synopsis of the process includes the stacked model architecture with two main stages; the LSTM encoder-decoder used for feature extraction and the LSTM NN for either RUL forecasting or binary classification following the maintenance request presented in section 2.3. It is worth emphasizing that the LSTM benchmark model do not include the autoencoder and feature are extracted based on the fuzzy logic detailed in section 2. The input to LSTM prediction model results from the encoder-decoder pre-trained to extract the best embedding representation from the multivariate time series input data. Especially, once the autoencoder is trained, the input data for the second stage of the model in then predicted from the embedded representation layer of autoencoder. For RUL forecasting the training is conducted by minimizing the mean absolute error MAE loss function $Loss = \sum_{i=m+1}^{S} \left\| \hat{y}^i - y^i \right\|$. The performance of the forecasting LSTM model is evaluated using the root mean square error (RSME) [17].For binary classification, the training is conducted by minimizing the binary cross entropy loss function. The evaluation is conducted based on standard binary classification metrics including the accuracy, the recall, the precision and the F1-score [5] [17]. The TP in the classification metrics refers to true positive values i.e., the number of correct data instances of the failure day correctly classified as failure. TN (True Negative) is the number of normal data instances (instances prior to the failure day) correctly diagnosed as normal and FP (False Positive) the number of normal data instances prior to failure incorrectly classify as failure and FN (False Negative) refers to number of failure instances mis-classify as normal data. In general, the accuracy is insensitive to imbalance data set and might produce misleading conclusions. Precision is a better measure of the capability of the model to correctly assign appropriate labels while the Recall evaluates the completeness of the model. F-score represents the balance between the two latter measures of the model.

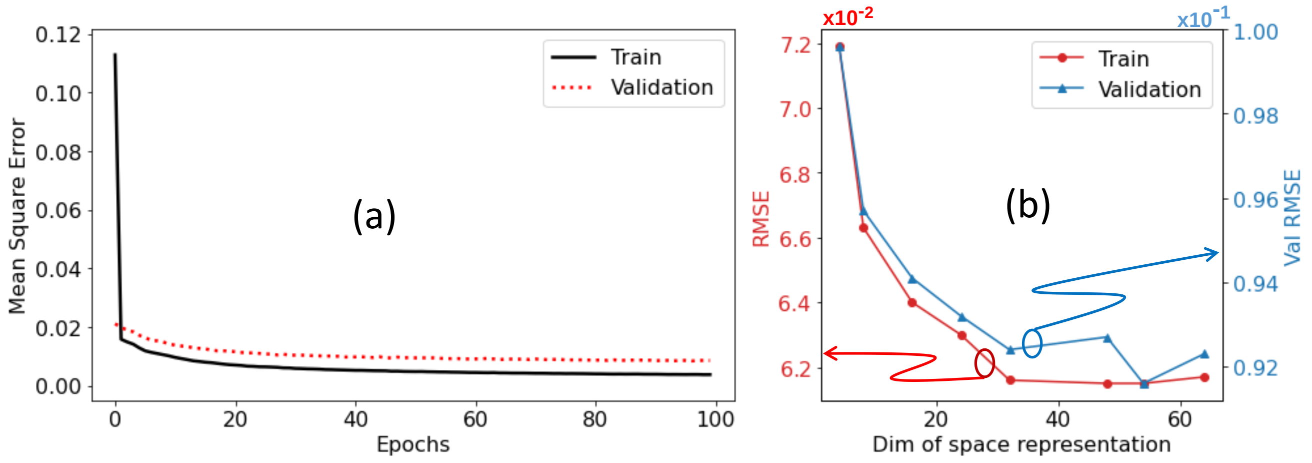


Figure 4: (a) Line plot of the train and validation loss from the LSTM autoencoder. (b) Evolution of train and validation RMSE versus the dimension of the space representation of the autoencoder.

IV. SLAGGING PREDICTION

To train the model, we define the train, validation and test set by splitting the data every three instances such that $(X,y)_{train} = \{x_t^i\,;\, y_t\}_{t=t_0,t_0+3,t_0+6,\dots}$ with $t_0 = 0,1,2$ for the train, validation and test sets respectively. Where y refers to the RUL or labels. We optimized the sliding window and obtained the optimal value of 12 i.e., we take every 12 consecutives instances to predict the RUL of the next instance. We keep a dropout of 0.2 in all the layers of both stages to avoid overfitting. Detail composition of the autoencoder model and the corresponding hyperparameter are provided in the appendix. The autoencoder is trained once on the train set and the dimensionality of the latent space is set to 32 as this value has shown to result to the best embedded representation after multiple tests. The embedded representation of the train, validation and test sets used to train, validate and evaluate the RUL forecaster or the label classifier is subsequently predicted from the trained autoencoder using the corresponding dataset. The significant decrease of the train loss and validation loss of the autoencoder (Fig. 4 (a)) illustrates the improvement of the encoding representation over the epochs. In order to find the best dimension of the space representation for the multivariate input data i.e., the best number features to train the forecaster, we trained the autoencoder multiple times with different space representation dimension. The train and validation RMSEs versus the space representation dimension reported in Fig. 4 (b) shows a rapid decrease toward an asymptotic minimum from a value 32. This finding suggests 32 features as good enough to efficiently represent the input data set and may open a relevant discussion regarding an eventual optimization of the data collection process.

A. RUL Inference

The parameters of the optimized model trained for RUL predictions are detailed in the appendix. The training is conducted using the encoded representation obtained from LSTM autoencoder. The Fig. 5 (a) presents the evolution of the MAE on both train and validation sets over the epochs. The Fig 5 (b) presents the visual comparison between true RUL values (section 2.3) and the predictions on the test set. The convergence of the MEA for both training and validation sets illustrates the absence of noticeable variance. The robustness of the approach is further supported by the value of the MSE and RMSE measures provided in Table 2 for both test and validation sets. Alternatively, the LSTM benchmark model with the same architecture as the forecaster was trained on the data set of the 47 selected features (see section 2.2). The model was first optimized on the train set and subsequently trained with the obtained best parameters (unit = 90, dropout=0, lr = 0.01). The obtained evaluation metrics for the same number of training epochs are provided in Table 2 (values between the parenthesis). The Fig. 5 (c) presents the comparison between the true RUL and the obtained prediction. One observes clearly that the stacked model trained with all features outperforms the simple optimized LSTM model trained on the 47 selected features.

	MAE	MSE	RMSE
Validation	0.196 (2.793)	0.169 (20.666)	0.411 (4.446)
Test	0.195 (2. 376)	0.215 (15.877)	0.464 (3.984)

Table 2. Evaluation metrics on the test and validation sets for the stacked (benchmark) models.

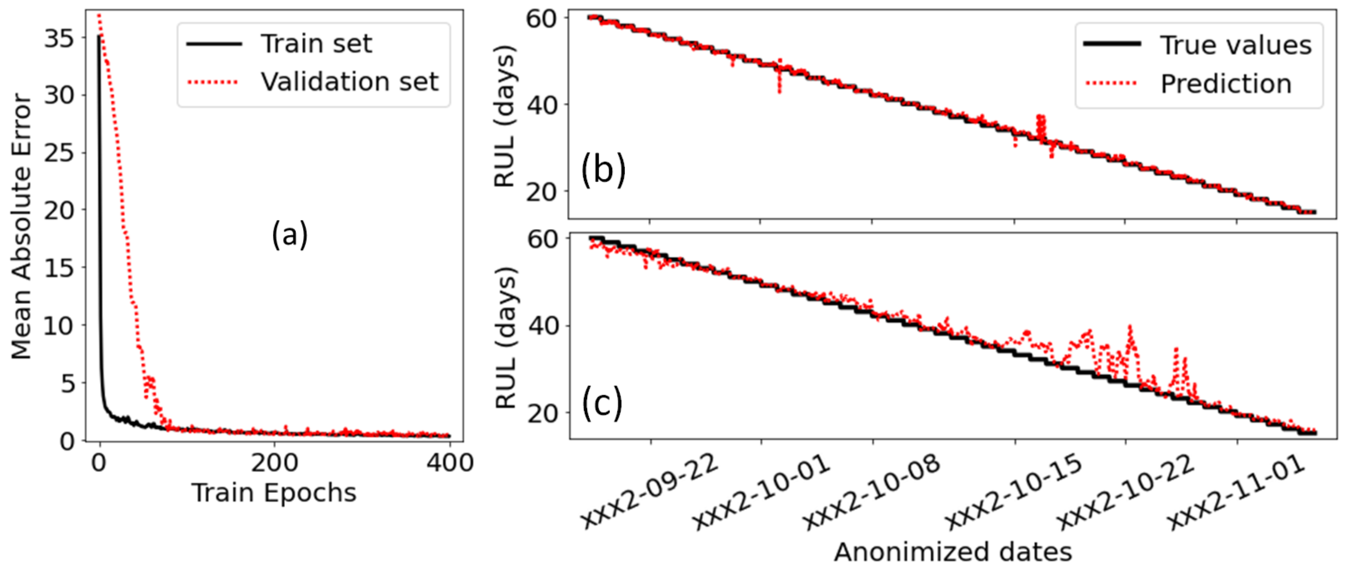


Figure 5: Results of the RUL forecasting. (a) Train and validation Loss over the epochs of the stacked model. (b) True and predicted RUL using the stacked model and benchmark model (c).

B. Classification Inference

The study for classification follows similar logic, i.e., we trained the stacked model on full data and a simple optimized LSTM model on the selected dataset. The architecture of the classifier in the stacked model is kept similar to that of the RUL forecaster (see appendix) and activation function is modified to sigmoid correspondingly to the classification purpose. However, while keeping the same configuration, the optimization of the simple LSTM model on the selected features leads to different hyperparameters (see appendix). More importantly, we implemented a class weight hyperparameter during the training of the classifiers which has proved to be extremely efficient in handling class imbalance. The weight assigned to each class corresponds to its proportion in the data set. We present in Fig. 6 the results for the classification request for both stacked model (top panels) and simple model (bottom panels). The left panel presents the confusion matrices of the prediction again the true label for the test set while the right panels present the visual comparison of true and predicted labels. We observe that both the stacked and benchmark models achieved more than 99% prediction accuracy on the test set as also confirmed on the right panel. We resume in Table 3 the values of different evaluation metrics on the test set for the stacked model (main data) and the simple model (data in parenthesis). One observes that the two models lead to efficient failure prediction on both validation and test sets. The confusion matrices show that the stacked model leads a single FN, classifying a normal instance as failure. The situation is reversed for the benchmark model which instead leads to a single FP. Interestingly, each misclassification takes place close to the transition point from normal to failure as observes on Fig. 7 (a & b) (right panels). Therefore, it has not major impact on the efficiency of the maintenance request. This shows that our LSTM models result to high performances in classifying failure instances and provide an efficient means to anticipate the slagging formation in coal power plant industry.

	Accuracy	**Precision**	**Recall**	**AUC**
Validation	0.9995 (0.9995)	0.9796 (0.9796)	1.0 (1.0)	0.9897 (0.9897)
Test	0.9991 (0.9995)	0.9792 (0.9796)	0.9792 (1.0)	0.9792 (0.9897)

Table 3. Classification metrics on the test set and validation set.

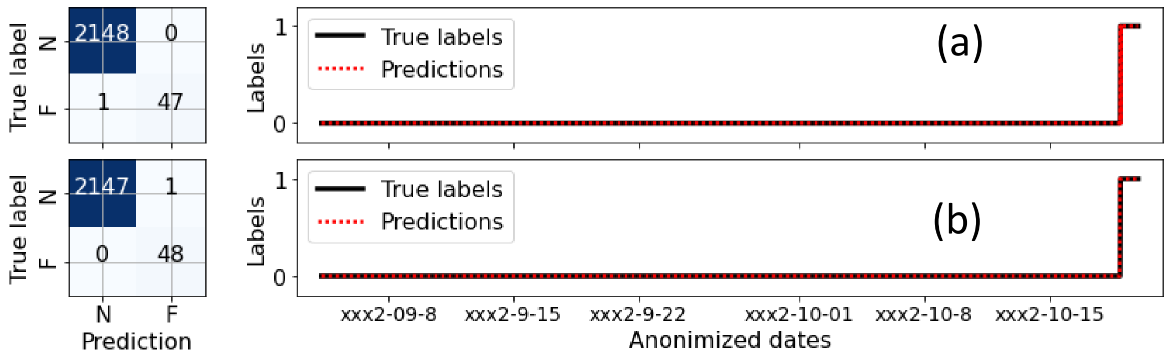


Figure 6: Results of the label classification. Confusion matrixes of the stacked model (top left panel) and benchmark model (bottom left panel). (a) True and predicted labels for the stacked model and benchmark model (b).

V. DISCUSSION AND FUTURE WORK

The stacked model demonstrates good prediction for both RUL forecasting and failure classification for the maintenance request elaborated in section 2.3. The simple model performs less for RUL forecasting while for classification it also demonstrates good performances. The comparative analysis of the results from LSTM autoencoder/LSTM stacked and benchmark models presented in section 4.1 and 4.2 shows that the autoencoder is not reducible to a simple dimension reduction tool. In contrast, the autoencoder is crucial for model performance particularly for RUL forecasting. Even for classification where both the stacked and benchmark models demonstrate high model performances; the stacked model is still preferable as it has shown to be more stable. This means that although the performance of the benchmark model can fluctuate from one training to the other, that of the stacked model remain similar. This demonstrates that in addition to efficiently reduce the dimension of the training data set, LSTM autoencoder is proved capable of extracting the hidden temporal pattern conducting to failure which subsequently enables a better learning of either the forecaster or the classifier.

The knowledge of a single slagging date was the one of the major challenges of the study. In order to set the conditions suitable to train and evaluate the RUL forecasting and classification models, we limited the study to two months historical data ending on a slagging date. We obtained high performance for both RUL forecasting and binary classification. Obviously, complete information would have enabled training the model on the complete data and likely enable improving its generality. We appeal data owners for more effort providing enough information when releasing data set for machine learning purposes. Furthermore, in the actual work, we consider the dataset from a single boiler to train the models due to the intrinsic constraints of the data set. However, an interesting step forward could concern implementing multiple boilers prediction by developing a single general model capable of prediction independently of the boilers. This requires knowledge of more slagging dates and could be implement following the approach developed by [16] for multiple sites forecasting using stacked model which has shown state of the art prediction.

VI. CONCLUSION

In this paper, we presented a machine learning approach for PdM applied to the challenge of slagging formation prediction. The approach is based on a stacked model consisting of LSTM autoencoder for feature extraction from high correlated multivariate time series data and LSTM NN for prediction. The predictor is trained either for both RUL forecasting or binary classification. A simple LSTM model is presented providing a means to gauge the impact of the autoencoder stage of the proposed stacked model. We developed in depth data analysis and feature extraction using the autoencoder opening route the optimization of collection methods. The stacked model demonstrates high performance for both RUL forecasting and binary classification. Although gingerly optimized the benchmark model less performs comparatively the stacked model for RUL forecasting. For classification, both models overall demonstrate comparable performances. However, the stacked model is observed to be more stable highlighting the importance of autoencoder. Valuable future works are discussed to move further the present study with the aim to develop a multiple sites prediction model that can be deployed for maintenance planification independently of the boiler.

APPENDIX: MODEL ARCHITECTURES AND HYPERPARAMETERS

LSTM autoencoder

Number of cells = LSTM(128)/LSTM(64)/TimeDistributed (Dense(32))/ LSTM(64)/ LSTM(128)
Hyper-parameters: dropout rate = 0.2

Layer (type)	Output Shape	Parameters
input_1 (Input Layer)	[(12, 114)]	0
lstm (LSTM)	(12, 128)	124416
lstm_1 (LSTM)	(12, 64)	49408

time_distributed	(12, 16)	1040
lstm_2 (LSTM)	(12, 64)	20736
lstm_3 (LSTM)	(12, 128)	98816
time_distributed_1	(12, 114)	14706

Total params: 309,122
Trainable params: 309,122
Non-trainable params: 0

Optimized LSTM predictor model trained either for RUL or Classification.
Hyper-parameters: dropout_rate = 0.2,
number_of_cells = 128/64/1
class_weight = {0:0.51, 1:23.02} (in the case of classification)

Layer (type)	Output Shape	Parameters
lstm (LSTM)	(12, 128)	74240
batch_normalization_1	(12, 128)	512
lstm_1 (LSTM)	(64)	49408
dense (Dense)	(1)	65

Total params: 124,225
Trainable params: 123,969
Non-trainable params: 256

Optimized LSTM model

a) RUL forecasting

Hyper-parameters: dropout_rate = 0,
number_of_cells = 90/90/1

Layer (type)	Output Shape	Parameters
lstm (LSTM)	(12, 90)	49680
batch_normalization_1	(12, 90)	360
lstm_1 (LSTM)	(90)	65160

Total params: 115,291
Trainable params: 115,111
Non-trainable params: 180

b) Classification
Hyper-parameters: dropout_rate = 0.1, number_of_cells = 128/32/1
class_weight = {0:0.51, 1:23.02}

Layer (type)	Output Shape	Parameters
lstm (LSTM)	(12, 128)	90112
batch_normalization_1	(12, 128)	512
lstm_1 (LSTM)	(32)	20608
dense (Dense)	(1)	33

Total params: 111,265
Trainable params: 111,009
Non-trainable params: 256

ACKNOWLEDGEMENT

The authors thanks ‘Energy de Portugal’ (EDP) for providing the data and more generally, his efforts of providing valuable data available on ‘EDP Open data’ repository to support efforts for in developing data-driven solution for various problems faced in industry 4.0. The link to the ‘EDP Open data’ repository is: https://opendata.edp.com/.

REFERENCES

[1] T. Zonta, ‘Predictive maintenance in the Industry 4.0: A systematic literature review’, *Industrial Engineering*, p. 17, 2020.
[2] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, ‘A systematic literature review of machine learning methods applied to predictive maintenance’, *Computers & Industrial Engineering*, vol. 137, p. 106024, Nov. 2019, doi: 10.1016/j.cie.2019.106024.
[3] Z. A. Bukhsh and I. Stipanovic, ‘Predictive Maintenance for Infrastructure Asset Management’, *IT Professional*, p. 6, 2020.
[4] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, ‘Machine Learning for Predictive Maintenance: A Multiple Classifier Approach’, *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, Jun. 2015, doi: 10.1109/TII.2014.2349359.
[5] Z. Allah Bukhsh, A. Saeed, I. Stipanovic, and A. G. Doree, ‘Predictive maintenance using tree-based classification techniques: A case of railway switches’, *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 35–54, Apr. 2019, doi: 10.1016/j.trc.2019.02.001.
[6] R. Prytz, S. Nowaczyk, T. Rögnvaldsson, and S. Byttner, ‘Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data’, *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 139–150, May 2015, doi: 10.1016/j.engappai.2015.02.009.
[7] T. dos Santos, F. J. T. E. Ferreira, J. M. Pires, and C. Damasio, ‘Stator winding short-circuit fault diagnosis in induction motors using random forest’, in *2017 IEEE International Electric Machines and Drives Conference (IEMDC)*, Miami, FL, USA, May 2017, pp. 1–8. doi: 10.1109/IEMDC.2017.8002350.
[8] T. Praveenkumar, M. Saimurugan, P. Krishnakumar, and K. I. Ramachandran, ‘Fault Diagnosis of Automobile Gearbox Based on Machine Learning Techniques’, *Procedia Engineering*, vol. 97, pp. 2092–2098, 2014, doi: 10.1016/j.proeng.2014.12.452.
[9] S. Butte, A. R. Prashanth, and S. Patil, ‘Machine Learning Based Predictive Maintenance Strategy: A Super Learning Approach with Deep Neural Networks’, in *2018 IEEE Workshop on Microelectronics and Electron Devices (WMED)*, Apr. 2018, pp. 1–5. doi: 10.1109/WMED.2018.8360836.
[10] K. Kulkarni, U. Devi, A. Sirighee, J. Hazra, and P. Rao, ‘Predictive Maintenance for Supermarket Refrigeration Systems Using Only Case Temperature Data’, in *2018 Annual American Control Conference (ACC)*, Milwaukee, WI, Jun. 2018, pp. 4640–4645. doi: 10.23919/ACC.2018.8431901.
[11] S. Choubey and G. P. Karmakar, ‘Artificial intelligence techniques and their application in oil and gas industry’, *Artif Intell Rev*, Nov. 2020, doi: 10.1007/s10462-020-09935-1.
[12] P. Xu, R. Du, and Z. Zhang, ‘Predicting pipeline leakage in petrochemical system through GAN and LSTM’, *Knowledge-Based Systems*, vol. 175, pp. 50–61, 2019, doi: https://doi.org/10.1016/j.knosys.2019.03.013.
[13] T. Plankenbühler, D. Müller, and J. Karl, ‘Influence of Fine Fuel Particles on Ash Deposition in Industrial-Scale Biomass Combustion: Experiments and Computational Fluid Dynamics Modeling’, *Energy Fuels*, vol. 33, no. 7, pp. 5911–5917, Jul. 2019, doi: 10.1021/acs.energyfuels.8b04200.
[14] M. U. Degereji *et al.*, ‘Predicting the slagging potential of co-fired coal with sewage sludge and wood biomass’, *Fuel*, vol. 108, pp. 550–556, Jun. 2013, doi: 10.1016/j.fuel.2012.12.030.
[15] M. Assaad, R. Boné, and H. Cardot, ‘A new boosting algorithm for improved time-series forecasting with recurrent neural networks’, *Information Fusion*, vol. 9, no. 1, pp. 41–55, Jan. 2008, doi: 10.1016/j.inffus.2006.10.009.
[16] L. Zhu and N. Laptev, ‘Deep and Confident Prediction for Time Series at Uber’, in *2017 IEEE International Conference on Data*

Mining Workshops (ICDMW), New Orleans, LA, Nov. 2017, pp. 103–110. doi: 10.1109/ICDMW.2017.19.

[17] H. D. Nguyen, K. P. Tran, S. Thomassey, and M. Hamad, ‘Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management’, *International Journal of Information Management*, vol. 57, p. 102282, Apr. 2021, doi: 10.1016/j.ijinfomgt.2020.102282.

[18] D. Bruneo and F. De Vita, ‘On the Use of LSTM Networks for Predictive Maintenance in Smart Industries’, in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, Washington, DC, USA, Jun. 2019, pp. 241–248. doi: 10.1109/SMARTCOMP.2019.00059.