PREDICTIVE MAINTENANCE FOR SMART INDUSTRY

A Thesis Submitted to
the Graduate School
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

In Computer Engineering

by

Asad ASADZADE

December 2020

İzmir

ACKNOWLEDGEMENTS

First of all, I would like to give my special thanks to my supervisor Assoc. Prof. Tolga AYAV for all kinds of support and guidance to make this study happen. Also, I am very grateful to him for sharing his time and knowledge with me.

I also would like to thank my friends for not leaving me alone and motivating me whenever I needed.

I am infinitely grateful to my family for their lifetime support including this stressful phase.

ABSTRACT

PREDICTIVE MAINTENANCE FOR SMART INDUSTRY

After the internet of things developed rapidly, it started to be used in many several industrial areas. Thanks to IoT, data that affect the health of any equipment or other important systems are collected. When these data are processed correctly, important information about the production process is obtained. For example, thanks to this data, systems based on machine learning are created to predict when various components will fail. Thus, maintenance operations are carried out before the component's breakdown, and replacement operations are performed if necessary. This strategy, called predictive maintenance, provides industries with advantages such as maximizing the life of components, reducing extra costs, and time saving.

In this study, we applied ARF method, which is based on stream learning, on Turbofan Engine Degradation Simulation Datasets which are provided by NASA to estimate the remaining useful lifetime of jet engines. As a result, we mentioned about the advantages of streaming learning over batch learning and compared our results with other batch learning based studies which are applied on the same datasets.

ÖZET

AKILLI ENDÜSTRİ İÇİN KESTİRİMCİ BAKIM

Nesnelerin interneti (IoT) hızla geliştikten sonra birçok endüstriyel alanda kullanılmaya başlandı. IoT sayesinde herhangi bir ekipmanın veya diğer önemli sistemlerin sağlığını etkileyen veriler toplanır. Bu veriler doğru işlendiğinde üretim süreci hakkında önemli bilgiler elde edilir. Örneğin, bu veriler sayesinde, çeşitli bileşenlerin ne zaman arızalanacağını tahmin etmek için makine öğrenimine dayalı sistemler oluşturulur. Böylelikle bileşen arızasından önce bakım işlemleri uygulanır ve gerekirse değiştirme işlemleri gerçekleştirilir. Kestirimci bakım adı verilen bu strateji, endüstrilere bileşenlerin ömrünü en üst düzeye çıkarma, ekstra maliyetleri düşürme ve zaman tasarrufu gibi avantajlar sağlar.

Bu çalışmada, jet motorlarının kalan faydalı ömürlerini tahmin etmek için NASA tarafından sağlanan 'Turbofan Engine Degradation Simulation Datasets' verisetleri üzerinde akış öğrenmeye dayalı Adaptive Random Forests yöntemini uyguladık. Sonuç olarak, toplu öğrenmeye göre akış öğrenmenin avantajlarından bahsettik ve sonuçlarımızı, aynı veri kümeleri üzerinde uygulanan diğer toplu öğrenmeye dayalı çalışmalarla karşılaştırdık.

To my family

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

ABBREVIATIONS

PdM: Predictive Maintenance

RUL: Remaining Useful Lifetime

RMSE: Root Mean Squared Error

ML: Machine Learning

CNN: Convolutional Neural Network

LSTM: Long Short-Term Memory

R2F: Run-to-failure

PdM: Predictive Maintenance

PvM: Preventive Maintenance

HT: Hoeffding Tree

HAT: Hoeffding Adaptive Tree

ADWIN: ADaptive WINdowing

ARF: Adaptive Random Forests

ARF-Reg: Adaptive Random Forests for Regression

MSE: Mean Squared Error

MAE: Mean Absolute Error

CHAPTER 1

INTRODUCTION

Industry started to advance rapidly in the 'Industry 4.0' era after the emergence and development of the Internet of Things technology. Today, a large amount of data is collected on several important sectors such as environmental science, industry etc. thanks to sensors and other devices. For example, using these sensors which are integrated into the equipment in factories, data such as working conditions of the equipment, heating and vibration are produced. The correct processing and analysis of these data helps in making important decisions for the production process. In addition, the health status of these equipment or more vital important systems are continuously monitored and analyzed with methods which based on collected data [7].

It is known how important the system, or any equipment maintenance activities are. Because maintenance activities directly affect the production process. Unnecessary and untimely maintenance activities may cause time wasting, extra cost and sometimes even life-threatening, most importantly. For this reason, maintenance activities must be carried out as planned and when it is necessary in order to provide advantages.

Recently, the Predictive Maintenance strategy has attracted more attention than other methods because it can address the above difficulties. In this strategy, the goal is to predict when maintenance will need to take place, using predictive tools such as Machine Learning based on data. And nowadays this strategy is used in many industrial areas [1].

Since Machine Learning (ML), Deep Learning and Artificial Intelligence are popular methods and effective approach, they are used for several kind of areas. There are experimental studies which are related with ML and Deep Learning used in predictive maintenance applications. Linear Regression, Support Vector Machines, Random Forests, Decision Trees, K-Nearest Neighbors, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) are some of them for both ML and Deep Learning methods. Carvalho et al. [7] handled literature review related with predictive maintenance with machine learning and showed 36 studies which based on batch learning and were published until 2019. Different kind of datasets were used in these studies for either classification or regression tasks. Kulkarni et al. [31] applied Random Forest

method to estimate presences of problems in refrigerators at any point as a classification task and as a result, they got 95% precision and 46% recall. In another study, Random Forest was applied on a dataset which include various sensor measurements related with cutting machine [32]. Also, in this study, early breakdowns of cutting machine were estimated as classification task and they achieve 95% overall accuracy. Biswal et al. [33] proposed Artificial Neural Network method to classify faulty and healthy wind turbine components and they obtained 92.6% classification accuracy. In another study [25] Long Short Term Memory was applied on engine dataset and they classify engine health condition to four class: fail, repair, caution and healthy. So, they got results with 85% accuracy. Dong et L. [35] used jet engine dataset which includes a set of sensor measurements in their study and applied LSTM-RNN and Naive Bayesian methods for regression task that aimed to estimate remaining useful lifetime of jet engines. In their study, LSTM-RNN achieved to get better result which RMSE equals to 17.84 than Naive Bayesian method. Kanawaday et al. [34] used four supervised models such as Naive Bayes, Support Vector Machine, Classification And Regression Tree and Deep Neural Network to forecast potential breakdowns and quality defects of slitting machine. As a result Deep Neural Network performed better than other three methods with 95.69% accuracy.

In following related studies, Turbofan Engine Degradation Simulation Datasets were handled which also we used too in our study to predict RUL of jet engines. In all these studies performance evaluations of methods were measured using Root Mean Squared Error (RMSE). It is important to point out that all these methods based on batch learning.

Table 1.1. shows RMSE values of methods used in each study. In the first study ten different ML algorithms were used on each four datasets. As a result, Random Forest performed better than other methods and best result was obtained from the first (FD001) dataset [1]. In the second study, LSTM was handled on only the first (FD001) dataset and as a result, RMSE value equals to 11.42 [2]. CNN algorithm was applied on each four datasets in third study [3]. The best result obtained from first dataset where RMSE value equals to 18.45. LSTM was used in the last study [4] as in the third, but on each four data sets not only FD001. On FD001 dataset, LSTM method performed better than on other datasets.

Table 1.1. RMSE values of methods in related studies.

Model	**FD001**	**FD002**	**FD003**	**FD004**
Random Forest [1]	24.95	29.64	30.55	33.79
LSTM [2]	11.42	-	-	-
CNN [3]	18.45	30.29	19.82	29.16
LSTM [4]	16.14	24.49	16.18	28.17

1.1. Goal and Contribution of the Thesis

Consider that IoT sensors collect large amount of data over time related with equipments and other system and this data are used to monitor and analysis of health condition of important systems. After a while, the data coming from the sensors create big data and thus it becomes difficult for batch learning methods to process such data. Also, in dynamic environments, the data changes over time, batch methods cannot keep up with new data and it may cause model breakdown. As a solution, batch methods need to be re-train on data that changes over time. But stream learning methods process each new sample one by one and update itself, so they can adapt to changing data over time.

The purpose of this thesis is to apply a Machine Learning method in order to estimates the Remaining Useful Lifetime (RUL) of jet engines. In this study, we handled the problem as a regression problem and used Turbofan Engine Degradation Simulation Datasets which are provided by NASA. Unlike the other studies which based on batch learning performed on these datasets, we used Adaptive Random Forests, which is a Machine Learning method adjusted for stream learning and showed that our results were not bad at all compared with batch learning methods.

1.2. Outline of Thesis

This thesis includes following chapters: Chapter 2 includes theoretical background of thesis such as Machine Learning, batch and streaming learning and their subtopics. Chapter 3 discusses datasets we used and our experimental results. And Chapter 4 includes ‘Conclusion’ and ‘Future Works’ parts of the thesis.

CHAPTER 2

THEORETICAL BACKGROUND

In this section, 'Maintenance Strategies', 'Machine Learning', 'Batch and Streaming Learning', 'Machine Learning for Streaming Data', 'Adaptive Random Forest Regression', 'Data Preprocessing', 'Evaluation Techniques' and concepts are mentioned.

2.1. Maintenance Strategies

With the development of technology in recent years, the field of maintenance in industry and other related fields has started to grow and develop rapidly. Maintenance is a strategy which combines all technical, managerial, and administrative activities that are intended to keep or restore an item in a state in which it can perform the required function throughout its life cycle. The purpose of maintenance actions is not only to minimize malfunctions or extend the life of an item, but also to minimize the operational costs associated with maintenance [5].

There are various nomenclatures for maintenance strategies in the literature. In this study [6], maintenance strategies are classified into three types:

1. *Run-to-failure (R2F) or Corrective Maintenance*: This is the maintenance technique applied after the equipment or any part has broken down or has been recognized. It aims to return this equipment to a state where it can fulfill its function. This is the simplest maintenance strategy. When there are unplanned breakdowns of the equipment, the situations such as repairing them and waiting for spare parts directly cause extra cost and time loss to factories.
2. *Preventive Maintenance (PvM)*: This strategy is a maintenance strategy that is carried out on a time-based or pre-planned basis. Overall, this strategy avoids potential equipment breakdown and is more advantageous than corrective maintenance. But sometimes insignificant maintenance or unnecessary parts change can cause extra cost to the factory.

3. *Predictive Maintenance (PdM)*: Using predictive tools such as ML techniques, this strategy predicts when maintenance should take place based on historical data of the equipments or parts. Highly accurate estimation of failures before they occur, eliminates problems such as time wasting, extra cost and operational safety. Figure2.1. shows general maintenance strategies and their aims. As we can see, each strategy has its own role.

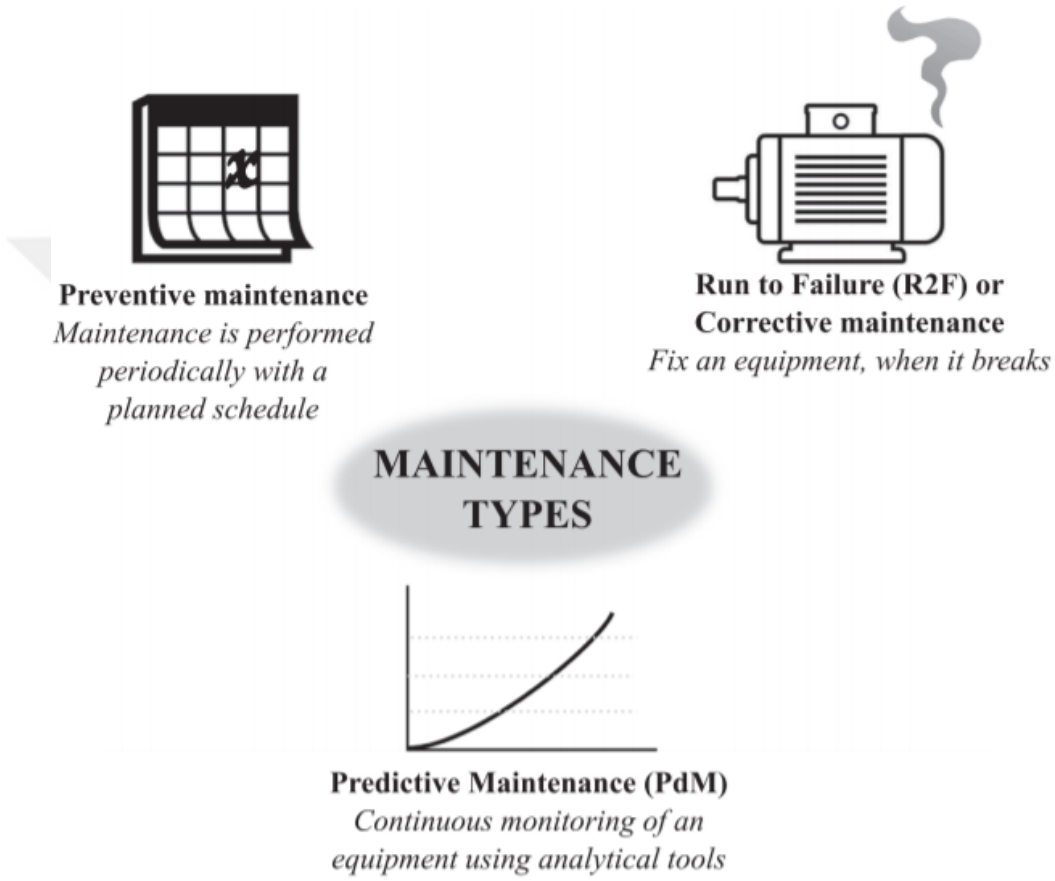

Figure 2.1. Maintenance strategies and their roles (Source: [7])

The act of maintenance is very important for industries. A good maintenance strategy should provide advantages such as improvement on equipment operating conditions, minimization of failures, reduction of maintenance costs, and maximization of equipment life. And thus, due to these advantages, PdM attracts great attention in the new industry 4.0 era. Nowadays PdM are applied to many industries such as oil and gas, manufacturing, rail, freight, shipping etc.

2.2. Machine Learning

As mentioned, previous section PdM strategy base on predictive tools such as Machine Learning (ML). So, using ML based predictive models it is possible to early estimate about equipment failures. ML which is a branch of Artificial Intelligence is one of the most popular topic and most used technology in several kind of areas nowadays. ML enables computers to learn from historical experience and make predictions about new incoming data accordingly. It means all ML models learn behaviors from historical data and this process known as 'model training' and data used this process is called 'training data'. After training process, model make prediction according to new data. It is very important to choose the most appropriate ML algorithm in order to make highly reliable predictions. At the same time, the data to be used should be carefully preprocessed before the training process. In general, ML models (for PdM applications) includes following minimum steps: *data selection, data preprocessing* and *model selection* steps.

- *Data collection* step defines how data is gathered for a ML model and how to select the most valuable data for this model.
- *Data preprocessing* step, we can divide this step into some steps such as data transformation step, feature selection step etc. In most cases, feature values in data sets have very different scales from each other, which makes the model mislead during prediction. Therefore, transformation rescales features from an interval to a new one and it is very important to build a model with higher accuracy. Also feature selection step is one of the most necessary steps to establish a highly reliable prediction model, it aims to select features which are the most effective for the target.
- *Model selection* step, this step the most appropriate ML model is selected for high accurate prediction for applications.

ML techniques are generally divided into the following three types: 'Supervised Learning', 'Unsupervised Learning' and 'Reinforcement Learning'. Figure 2.2. illustrates these ML techniques.

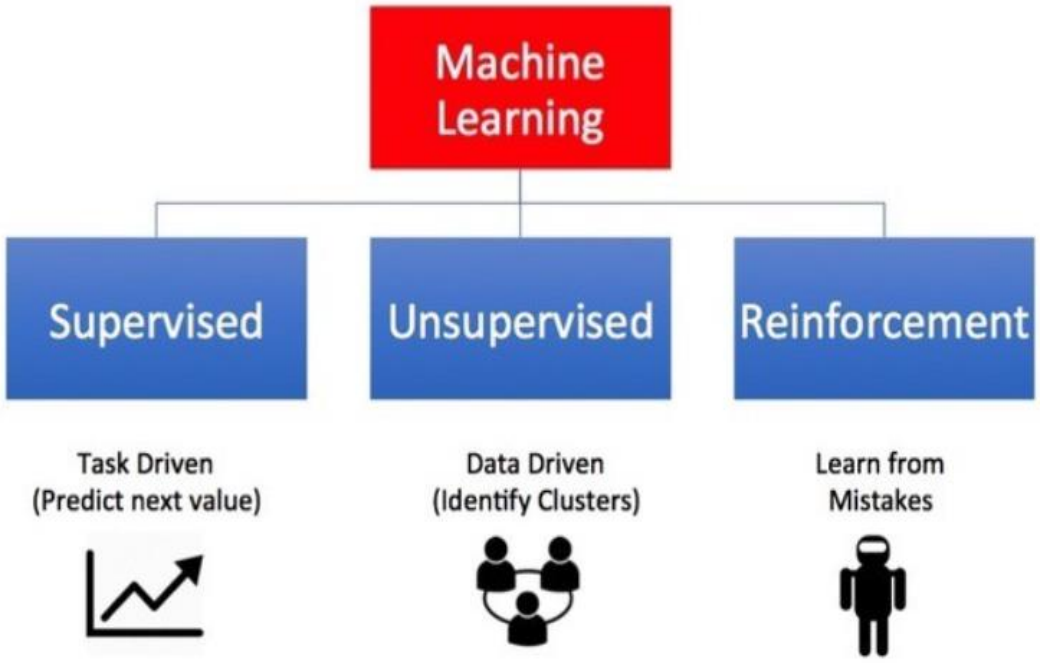

Figure 2.2. Machine Learning techniques (Source: medium.com).

2.2.1. Supervised Learning

Supervised learning is a ML technique that requires each training instance with it's corresponding target value. In simpler terms, each training sample comes as both input and output values. And after training process, the model predicts the output value which means the target value for a new input. One of the important steps to build a high accuracy ML model is to prepare the data correctly to train this model.

Supervised Learning is divided into ‘Classification’ and ‘Regression’ according to the problems posed. In the classification task, the target/output values in the data set consist of classes, that is, categories. So, in the training process, the models are trained on information which the input values are belongs to defined classes. As a result, models make predictions that any new input values belong to any class. Generally, there are two types of classification problems as binary and multiclass classification. In binary classification, classification tasks are handled which have two class labels. Email spam detection tasks may be an example for binary classification tasks. The outputs of these models are as ‘spam’ or ‘not spam’. But unlike binary classification, in multiclass classification tasks datasets include samples which belongs to more than two class. As an example, the face classification task in face recognition systems. In these systems,

models estimate photo belongs to which face among thousands of faces.

Regression tasks are also Supervised Learning method but difference between Classification and Regression is the results of prediction which in Regression they are numeric values but in Classification they are categoric values [8]. There are many popular ML algorithms for Supervised Learning tasks. Some of them are Logistic Regression, K-Nearest Neighbors, Decision Trees, Support Vector Machine, Naive Bayes, Random Forest etc.

2.2.2. Unsupervised Learning

Unlike supervised learning, unsupervised learning requires training data without labels in the training process. In contrast to supervised learning, models in unsupervised learning do not need to be supervised by users. This learning technique allows models to learn by themselves structures and knowledge which were not defined before. Unsupervised Learning also known as clustering problem. In this type of clustering problems, dataset is divided into groups considering the similarities between the values in the data set. Data in each different group has the same properties. In this way, the properties of the new incoming data are considered and this data are assigned to the clusters to which it is closest. Clustering is used in many real-world problems such as market segmentation analysis, grouping plants according to their characteristics in biology, classify people according to behaviors in twitter.

2.2.3. Reinforcement Learning

Reinforcement Learning is one of the ML techniques which makes enable systems learn from behaviors and uses feedbacks during learning process. It means, Reinforcement Learning aims to find the target path or behavior. If it can find correctly, reward signal arises, if cannot find penalty signal arise and system tries to avoid actions which cause penalty signal. So, the goal in Reinforcement Learning systems is to maximize rewards.

2.3. Batch and Streaming Learning

Data collection and processing step is the first step of a ML application. In generally, data is gathered and processed as batch before learning process. Although this method is a new technology in various applications, it is not suitable to handle the data as batch for streaming data. In order to process data in batch learning methods, it must be sufficiently large and accessible. But in the evolving data stream methods, the data are handled one by one and can continue endlessly, so it is not appropriate to store the data in anywhere. In addition, since the data is constantly changing in dynamic environments, the above steps should be repeated multiple times in order to adapt the established model to new data and to ensure maximum performance. For example, adaptation is very important in the following sample applications based on rapidly changing environments [9].

Financial markets: for example, the New York Stock Exchange collects about 1 terabyte of data per day. And these data are subject to changes due to the condition of the markets and the influence of different factors. As a result of this change, old data becomes useless for established prediction models. Predictive models must adapt quickly to new incoming data in such fast-changing environments.

Predictive maintenance: We know how important IoT is nowadays and produces very valuable data. Since these data can be related to the health conditions of aircraft, cars and other important systems, and it is very important that the established models adapt to changing environments in order to make an accurate prediction of degradation.

Supply chain: Thanks to advanced technologies, many sectors now use automated systems for their supply chains. But sometimes these automatic systems cannot cope efficiently with product demand. With the beginning of the COVID-19 pandemic, in the first week, products related to the pandemic such as face mask filled the 10 most searched words on Amazon.

Climate change: We know that how big and important environmental science data is. For example, the NASA holds about 24 petabytes of data related with environmental science. The importance of environmental science data can be explained in this way: For example, food production is seriously affected by climate change, or deterioration of the water cycle can cause heavy rains, which means a big flood risk. Today, IoT devices are capable of collecting rapidly changing environmental data, and ML methods should be able to

cope with such changes. As can be seen from the examples for the dynamic environments above, batch learning systems cannot easily adapt to rapidly changing environments, which causes problems such as model degradations. In general, in batch learning systems, users decide to train the model after sufficient data is collected. Hence, two major problems arise. First, models based on batch learning cannot incorporate new incoming data into the process. Thus, these models require all steps to be rerun as new data is collected. The second is to decide whether the model should be trained according to new data only. In general, the value of variation on the data is taken into account before retrain the model on only new data. But in stream learning systems, models update themselves according to each new data and can adapt to such changes [9].

Figure 2.3. shows the relationship between predictive performance and investment in resources such as time and memory for both batch and stream learning.

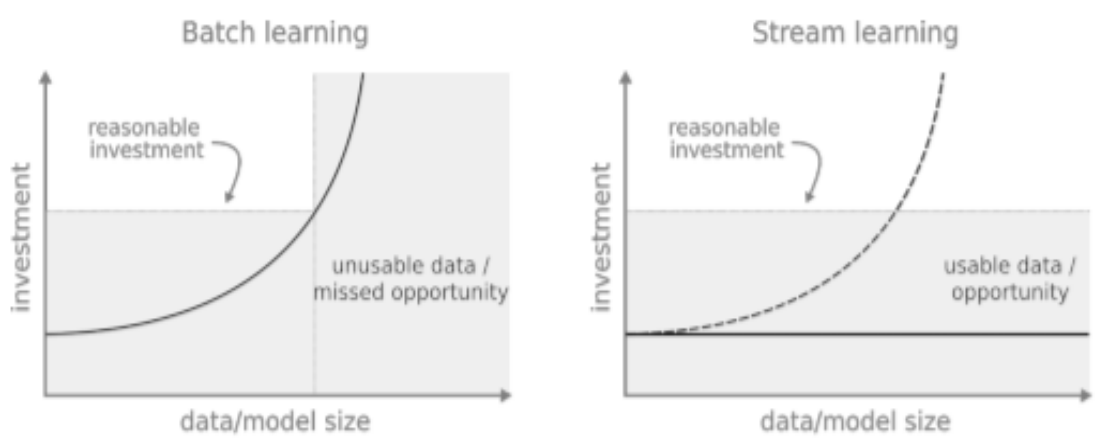

Figure 2.3. Investment in time and memory for batch and stream learning (Source: [9]).

As we can see that investment on the batch learning are increasing while data is increasing but investment is constant despite data increase in stream learning. It is very important that models based on stream learning not only get high performance but can also process infinite incoming data stream. In stream learning, incoming data should be processed as fast as possible, and the model must be ready to process the next incoming data.

2.4. Machine Learning for Streaming Data

In general, supervised learning for streaming data is described as follows [28].

$$A = \{(\vec{x}_t, y_t)\} | t = 1, \dots, T \quad (1)$$

Where T→ ∞, $\vec{x}_t$ is feature value and y_t its target value. The values of y_t are continues in regression tasks but in classification task it is categorical. Here the goal is to predict the target y value corresponding to each new incoming $\vec{x}$ value.

Unlike batch learning which evaluates all data in the training process, the data are handled one by one during training process and models is updated with each new data sample $(\vec{x}_t, y_t)$ in the stream learning. Different from batch learning models, models based on stream learning must fulfill following requirements [10].

1. Only one sample should be processed at a time and it only should be handled one time.
2. Limited memory should be used. Because data streams can continue endlessly, it is not practical to store them.
3. Should be worked in a limited time. Delayed operations which could cause the algorithm to fail should be avoided.
4. Models based on stream learning must be constantly updated and always ready to predict

Repeated cycle in Figure 2.4. shows the use of a stream learning algorithm and the steps in which the requirements should be done. In the 1st step algorithm receives the available sample from the stream by fulfilling Requirement 1. In the second step, the received sample processed by the algorithm and this step the algorithm must fulfill Requirement 2 and Requirement 3. In the last step the algorithm makes prediction and must be able to ready to receive next sample (Requirement 4). Thus, stream learning algorithms must fulfill these mentioned important requirements in order to process streaming data which continuously income to model.

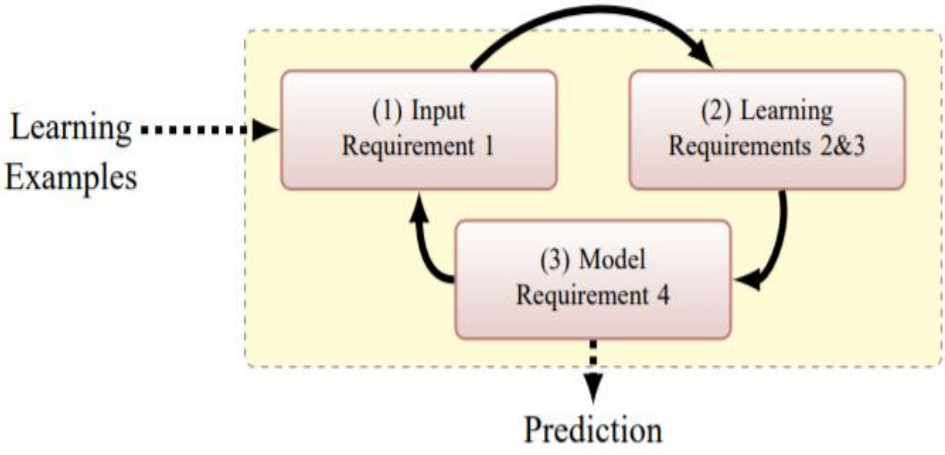

Figure 2.4. The stream learning algorithm cycle (Source: [10]).

2.4.1. Concept Drift

The concept drift in the field of ML is known as the change in the relationship between features and targets over time. Changes in *p(y|X)* are known as real concept drift and this type of drift cannot be affected changes which happens in *p(X)*. Generally, in the stream learning concept drift defined as following:

$$p_{t_0}(X, y) \neq p_{t_1}(X, y) \qquad (2)$$

Where p_{t_0} and p_{t_1} indicate joint distributions between X features and y targets at t_0 and t_1 time points respectively [11].

Over time, changes in data distributions i.e., concept drift, may occur in different ways. Figure 2.5. illustrates these different types of changes.

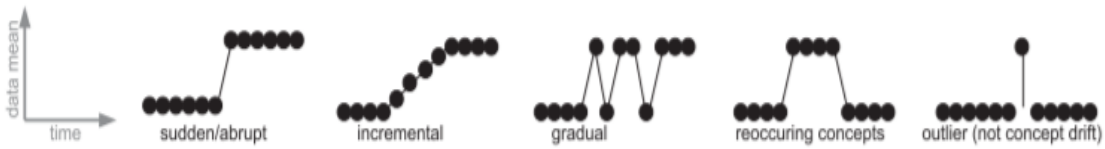

Figure 2.5. Distribution changes over time (Source: [11]).

- **Sudden/Abrupt** drift occurs when suddenly switching from a concept to another concept.
- **Incremental:** unlike sudden/abrupt, there is no sudden transition between concepts, that is, intermediate concepts arise while the transition.
- **Gradual:** there are random examples belong to both concepts occur during the transition.
- **Reoccuring:** again, reoccuring old concepts sometimes during the data streaming is continues**.**
- **Outlier:** it is important point out that outliers is not concept drift because generally they are far from the mean of the samples in the dataset and do not continue during data stream.

Since methods based on stream learning constantly update themselves so they can adapt to concept drifts. But despite this, several methods are recommended in the literature to detect and react fast and efficiently concept drifts. For example, the Hoeffding Tree (HT) [12] and Hoeffding Adaptive Tree (HAT) [13] algorithms which both are decision tree and are set for stream learning. Difference between these two algorithms is that the first one does not have any special property to detect concept drifts but the second can detect concept drifts using ADaptive WINdowing (ADWIN) [14].

ADWIN, one of the most common drift detection methods, uses windows (W0, W1) on the last incoming samples to determine if there is a change in distributions. The average values of the samples in W0 and W1 are compared with the distribution of the new observed sample to determine whether there is a change , so if the equality is broken, a warning signal emerges it means that the drift has been detected so W0 is changed with W1 and a new W1 window is launched.

2.4.2. Effect of Concept Drift on Stream Learning

If the concept drift is not handled correctly, it can seriously affect the performance of the predictive model. Stream learning models are able to adapt to concept drifts because they update themselves constantly, but models which based on batch learning cannot cope with the concept drifts when these drifts occur because the models are trained with different dataset (train data) at first.

In following example, stream learning classification task is handled using two

classification model namely HoeffdingTreeClassifier (HT) and HoeffdingAdaptiveTree (HAT)Classifier [15]. HAT is adaptive version of HT it means HAT have drift detection property (uses ADWIN) unlike HT. Used data in this study includes three gradual drifts at 5000, 10000 and 15000 points. Figure 2.6. illustrates the performance of HT and HAT and their reactions to concept drifts on this dataset.

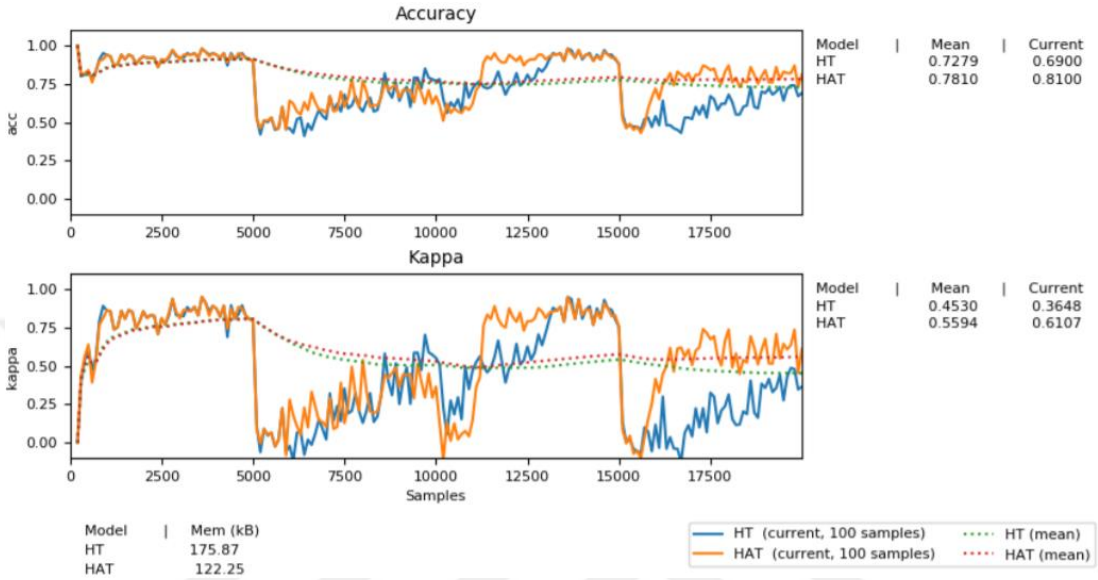

Figure 2.6. Comparison of HT and HAT on drifts (Source: [15]).

As seen from Figure6, both models show almost the same performance at the beginning. But after 5000 and 15000 samples, drifting occurs and the performance of both models decreases, but the HAT recovers faster. We can also see that after 1000 sample HT performs better that HAT, but it doesn't take long. Also, it is observed that HAT is more effective against drifts as well as better than HT in overall performance.

2.4.3. Adaptive Random Forests

Random Forests is one of most popular algorithms used in classification and regression problems which based on batch learning [16]. Random Forests is ensemble method which is combinations of many decision trees that are able to make prediction separately. Random Forests uses the bagging method [17] to create decision trees to avoid overfitting and selects features randomly while each node split in the trees. Lots of

passes over input data are required to implement bootstraps for trees which are combined in Random Forests. But it is not practicable to multiple passes over input data in stream learning. Therefore, the following requirements must be fulfilled for random forests can be applied to such data streams. First, online bagging process must be used, second, a limited feature split must be used in each leaf split. To achieve the second requirement, base trees must be modify, feature set should be restricted effectively to a random m subset for more splits (m < M, M is total number of features). To adaptation to first requirement, new bagging method was proposed namely 'leveraging bagging' [18].

Adaptive Random Forests (ARF) is the adaptive version of Random Forests for streaming learning. In general models which based on stream learning can adapt the concept drifts because they update themselves continuously. However, some stream learning methods use extra drift detection methods. For example, ARF uses ADWIN drift detection method as a default. In addition, ARF uses Hoeffding trees [19] as base learner.

Hyper-parameter tuning which aims to find optimum parameter values for models is also very important for ML models to reach high accuracy. There are several hyper-parameter tuning methods for batch learning based models in literature. Whereas it is very difficult to apply this approach for stream learning based models. The naive solution to this challenge is to separate the first incoming samples from the stream and find optimum parameter values on this subset. But it is debatable that how this approach is effective [20]. One more advantage of ARF is that it achieves highly reliable results without further parameter tuning. Generally, ARF has following limited main parameters: m (feature size), n (number of trees) and a parameter which control drift detection method [18].

2.4.4. Data Preprocessing

Data preprocessing is an important aspect for ML tasks because it directly affects performance of ML models. Data preprocessing includes some steps such as Data Transformation step which rescales features into defined range and Feature Selection step which select the most relevant features related with learning task. Generally, in batch learning, data preprocessing, model training and testing are handled as separate steps and respectively. So, according to the result of this processes, future data are processed.

The same processes above are also required for stream learning. But unlike batch learning, stream learning requires these steps continuously during data streaming. Whereas it is very complicated to make these adjustments for streaming learning. For example, rescaling a feature at a certain range in streaming data is a very complex process because the statistical information of the data such as *max* and *min* is not known priori because new data are constantly coming. So preprocessing operations are handled as offline in most of streaming learning task due to lack of online preprocessing studies for streaming data [20].

2.4.5. Feature Selection

In supervised learning handled datasets may include multiple features depend on problems field. Some of these features don't capture sufficient information about targets. Such irrelevant and unnecessary features can cause the prediction model to be misled during training process and at the same time reduce the overall performance of this model. At the same time these unrelated features increase the time wasting and extra computation requirements. In order to avoid such challenges, Feature Selection is important for ML tasks. Feature Selection aims to select the most relevant features for the learning process. Generally, Feature Selection algorithms are classified to following methods [29]:

Filter: In this method, statistical measurements are used to score each feature and each feature is ranked according to this scoring. So, the most related features are selected by this way. It is important to point out that, in this method Feature Selection takes place before the learning process and independently of the model. Figure 2.7. shows Filter feature selection method.

Figure 2.7. Filter feature selection method (Source: [21]).

Wrapper: In this method, firstly a features subset is selected for train model and then new features are added to or previous features are removed from the subset until to find appropriate subset which include the most related features. The disadvantage of this method is that it requires a lot of computation. Figure 2.8. illustrates Wrapper feature selection method.

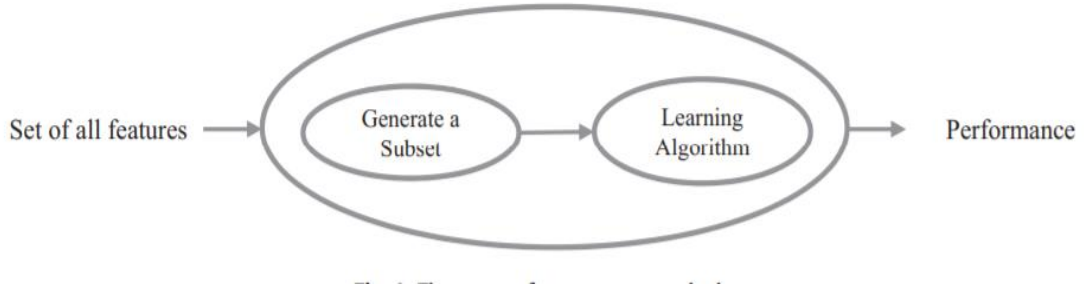

Figure 2.8. Wrapper feature selection method (Source: [21]).

Embedded: In this method, the most relevant feature subset search happens during learning process. The purpose of this method is to find the features which are the most effective on model accuracy when the model is created. Figure 2.9. depicts Embedded feature selection method.

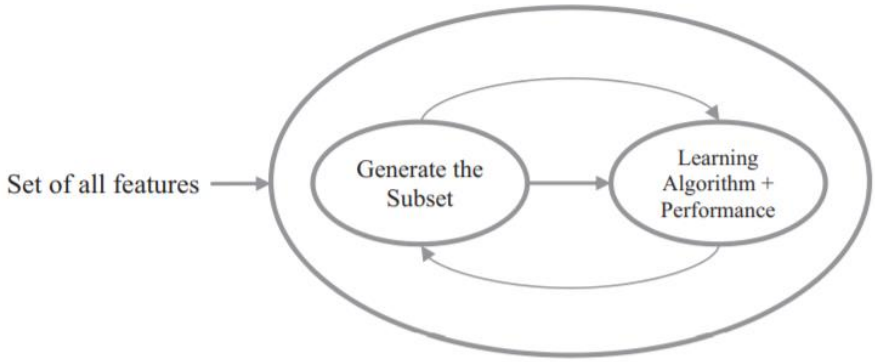

Figure 2.9. Embedded feature selection method (Source: [21]).

Filter methods are more efficient than wrapper and embedded methods against overfitting as well as faster than other two methods [22].

2.4.6. Evaluation Techniques

Due to the infinite nature of the streaming data, model evaluation methods such as train test set split and cross validation which are used for models created on batch data are not suitable for models which created on the streaming data. So, following special performance evaluation methods are suggested for predictive models created for streaming data [23]:

Holdout or Periodic Holdout: in this method, some amount of data is predefined from the stream for test the model and after these seen test data or after a while predefined testing samples are renewed again. So, in this method, only defined samples are used to test the model not all samples from the stream.

Prequential Evaluation (PE): This method allows all the samples in the stream to be tested first and then used to train the model after updating the model. Thus, the model is tested on samples that it has not seen before. In addition, the model does not need predefined test data specifically for testing. All data are used to test-then-train the models. This makes the models make more accurate prediction because models update continuously.

CHAPTER 3

EXPERIMENTS AND EVALUATIONS

3.1. Used Technologies

Experiments part of this study has been carried out on Jupyter Notebook using python program language and libraries such as numpy, pandas, seaborn, scikit-learn which are necessary for ML and scikit-multiflow library for streaming learning.

Python programming language is a general-purpose language like other programming languages like C ++, JavaScript, C # and Java. But in recent years, the advantages of python compared to other languages have further fueled interest in it. For example, to be easier and flexible, to be more standard than other competitors in the field of Data Science, to work in almost all operating systems, to have a large number of libraries and to be free.

Jupyter Notebook is a web-based technology and allows us to write our cod in independent cells. Moreover, using Jupiter Notebook, we can easily implement processes such as data cleaning, statistical modeling, data visualization, ML etc.

Numpy is the core library which is known as Numerical Python. Numpy library helps us to do mathematical operations on multidimensional arrays and matrices easily.

Pandas is one of the most popular python libraries to use in analysis of data which backend source code was strictly written in C or Python, providing highly optimized performance.

Seaborn is the widely used data visualization library which based on matplotlib. Seaborn library offers to draw high-level, useful and easy interpretable graphics.

Scikit-learn is an open source Python library that uses a single framework to implement a variety ofML, pre-processing, cross-validation and visualization algorithms.

Scikit-multiflow is a ML library that was made for multi-output and stream data and was written in Python. It is necessary to point out that scikit-multiflow and scikit-learn which is designed for batch learning are different from each other.

3.2. Dataset Description

The dataset namely “Turbofan Engine Degradation Simulation Dataset” handled in this study is taken from the Nasa data repository. C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) tool was used which simulates realistic large commercial turbofan engine deterioration while collecting data. This software was developed using MATLAB and Simulink [24]. Figure 3.1. illustrates the basic parts of the engine simulation.

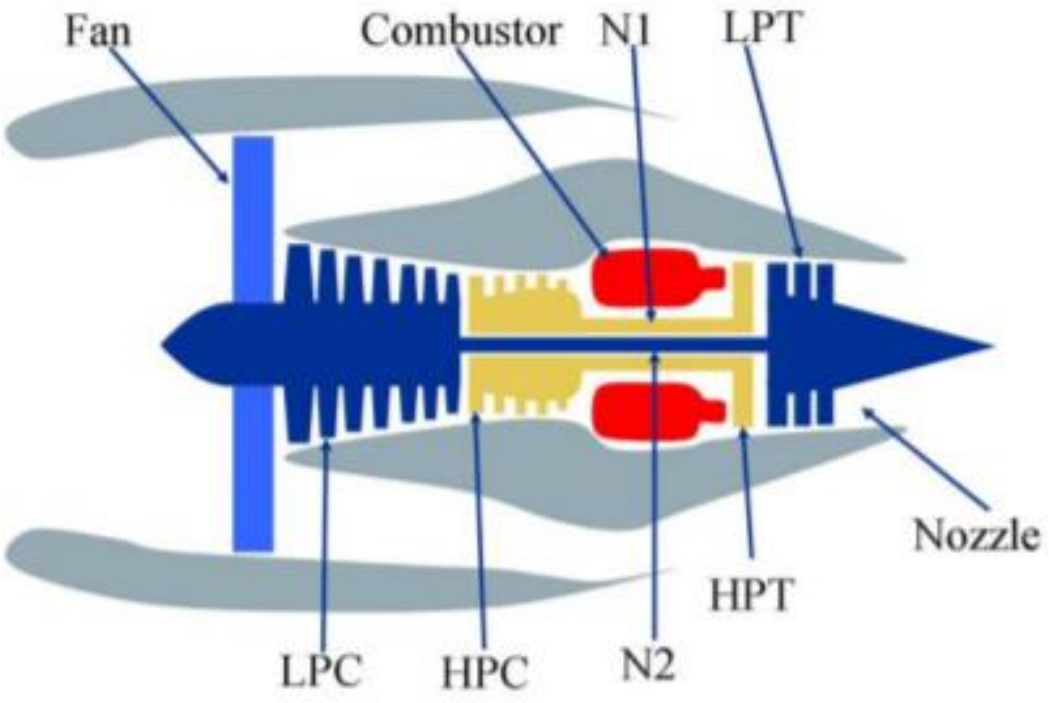

Figure 3.1. Simplified diagram of engine simulated in C-MAPSS (Source: [24])

There are four datasets and they differ from each other according to their operational settings. Each one has training, testing and RUL sets in the repository. In this study, all datasets (FD001, FD002, FD003, FD004) are used for experiments. Each dataset contains IDs for each different engine, 21 different sensor measurements, data describing three ambient conditions of machines and the working time as a cycle. Table 3.1. contains description of dataset and helps to understand it better [25].

Table 3.1. Description of the dataset (Source: [25]).

ID	ID of Engine
Cycle	Time, in Cycle
Setting1	Altitude
Setting2	Mach Number
Setting3	Throttle resolver angle
Sensor1	Total temperature at fan inlet (OR)
Sensor2	Total temperature at LPC outlet (OR)
Sensor3	Total temperature at HPC outlet (OR)
Sensor4	Total temperature at LPT outlet (OR)
Sensor5	Pressure at fan inlet (psia)
Sensor6	Total pressure in bypass-duct (psia)
Sensor7	Total pressure at HPC outlet(psia)
Sensor8	Physical fan speed (rpm)
Sensor9	Physical core speed (rpm)
Sensor10	Engine pressure ratio (P50/P2)
Sensor11	Engine pressure ratio (P50/P2)
Sensor12	Ratio of fuel flow to Ps30 (pps/psi)
Sensor13	Corrected fan speed (rpm)
Sensor14	Corrected core speed (rpm)
Sensor15	Bypass Ratio
Sensor16	Bumer fuel-air ratio
Sensor17	Bleed Enthalpy
Sensor18	Demanded fan speed (rpm)
Sensor19	Demanded corrected fan speed (rpm)
Sensor20	HPT coolant bleed (lbrn/s)
Sensor21	LPT coolant bleed (lbm/s)

Engines run normally at the beginning of the time series, but after some cycles over time, the motors start to malfunction and fail suddenly. Each of four datasets contains train, testing and ground true sub datasets.

- Train dataset is engines' run-to-failure data.
- Test dataset contains without failure records.
- Ground True dataset consist of information that related with true remaining cycles for each engine in the testing data.

In Figure 3.2. we can see first and last five rows of train set of FD001 dataset which includes 20631 samples (rows) and 26 features (columns) in the train set.

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	100	196	-0.0004	-0.0003	100.0	518.67	643.49	1597.98	1428.63	14.62	...	2388.26	8137.60	8.4956	0.03	397	2388	100.0	38.49	22.9735
20627	100	197	-0.0016	-0.0005	100.0	518.67	643.54	1604.50	1433.58	14.62	...	2388.22	8136.50	8.5139	0.03	395	2388	100.0	38.30	23.1594
20628	100	198	0.0004	0.0000	100.0	518.67	643.42	1602.46	1428.18	14.62	...	2388.24	8141.05	8.5646	0.03	398	2388	100.0	38.44	22.9333
20629	100	199	-0.0011	0.0003	100.0	518.67	643.23	1605.26	1426.53	14.62	...	2388.23	8139.29	8.5389	0.03	395	2388	100.0	38.29	23.0640
20630	100	200	-0.0032	-0.0005	100.0	518.67	643.85	1600.38	1432.14	14.62	...	2388.26	8137.33	8.5036	0.03	396	2388	100.0	38.37	23.0522

Figure 3.2. The first and last five rows of train dataset (FD001)

Figure 3.3. (a,b,c,d,e,f) shows six different sensor readings as line plots for an engine which its ID number is 1 in the FD001 dataset. As we understand, each plot shows a line that causes the first engine to malfunction. As understood from the figure, the readings of sensor1 and sensor16 do not change from the beginning to the end. Unlike this, the values of sensor4 and sensor11 increase, while the values of sensor7 and sensor14 decrease and they cause degradation of the engine.

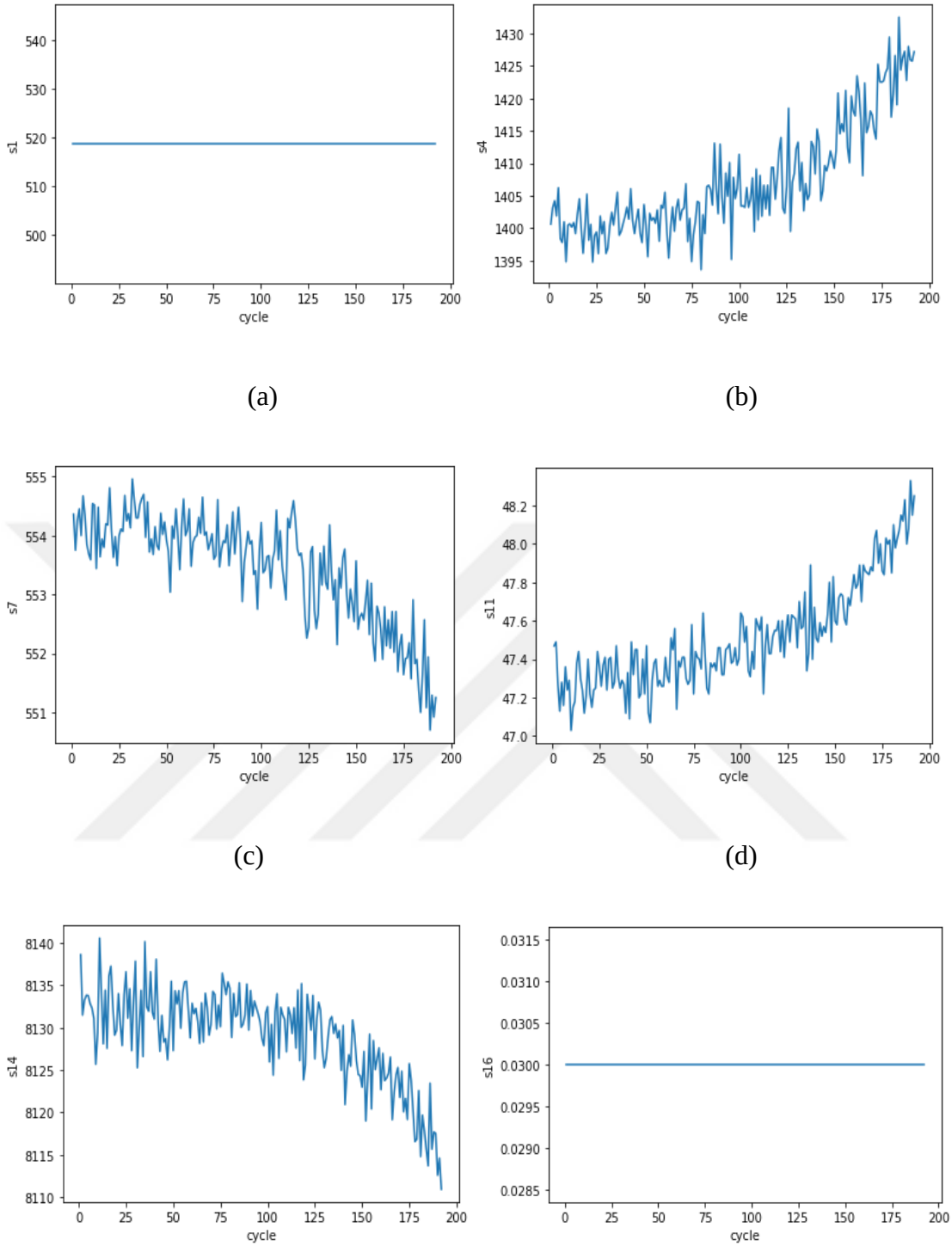

Figure 3.3. (a,b,c,d,e,f) Six sensor readings until engine 1 fails.

3.3. Data Preprocessing

As we know, each of FD001, FD002, FD003 and FD004 consists of train and test sets separately, and we handle whole dataset in this study by combining the train and test sets in each of them. Table 3.2. shows the number of samples in each and their sum respectively.

Table 3.2. Number of samples in each dataset.

Datasets	**Train set**	**Test set**	**Total**
FD001	20631	13096	33727
FD002	53759	33991	87750
FD003	24720	16596	41316
FD004	61249	41214	102463

Training and test sets consist of id, cycle, operational settings and sensors columns. In order to use these data for streaming models, we first need to calculate the real RUL values in each cycle and add these values as columns to the dataset. Following equation helps to understand better. RUL is remining number of cycles until engine will fail

$$RUL_{\text{i}} = last_{cycle} - Cycle_{\text{i}} \quad (3)$$

where *last_cycle* is the last cycle that is taken from Cycle feature in the dataset. After adding RUL values, we drop ID feature, which is just ID number and identify engine, because it is not informative for prediction process and does not affect RUL.

3.3.1. Data Transformation

In most cases, feature values in data sets have very different scales from each other, which makes the model mislead during prediction. Therefore, transformation is very important to build a model with higher accuracy. In this study, we used the Min-Max transformation method to rescale features in [0-1] range. Min-Max transformation rescales features from an interval to a new one. Following equation is used for this method.

$$x'_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{4}$$

where x_i is input instance, $\min(x)$ and $\max(x)$ are minimum and maximum values and x' is transformed data corresponding to the x.

3.3.2. Sensor Selection (Feature Selection)

In order to establish a highly reliable prediction model, it is very important to select the sensors that characterize well the health condition of the engine. As we know, there are 21 sensor values in the dataset we used and some of these sensors contain very little information about engine deterioration. Also, there are some sensors whose values do not change at all during the process and these sensors are not useful for RUL estimation and are considered useless to be used [26]. In this study we have made feature selection operation using Pearson's Correlation Coefficient method which based on Filter feature selection. This method is represented with following formula.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{5}$$

where $\rho_{X,Y}$ is correlation coefficient of X and Y, $cov(X,Y)$ is covariance of X and Y, $\sigma_X \sigma_Y$ is standard deviation of X and Y respectively.

As a result, when ten sensors for FD001, FD002, FD004 datasets and twelve sensors for FD003 dataset were selected which most characterize the engine breakdown, we achieved more accurate results. Table 3.3. shows selected sensors for each dataset. In

other words, we tried multiple combinations to determine the optimum number of sensors for each data set and got the best result in the above case.

Table 3.3. Selected sensor for each four datasets.

Dataset	**Selected Sensors**
FD001	s2, s3, s4, s7, s11, s12, s15, s17, s20, s21
FD002	s3, s4, s9, s11, s14, s15, s16, s17, s20, s21
FD003	s2, s3, s4, s7, s8, s9, s10, s11, s12, s13, s14, s17
FD004	s2, s3, s4, s9, s10, s11, s14, s15, s16, s17

3.4. Experimental Results

Experimental results are discussed in this section. ARF-Reg (a version of ARF which is proposed for regression tasks [30]) algorithm was applied on each four datasets that we explained in the previous section. There are several methods for evaluating the performance of established models in stream learning. In this study, we used Prequential Evaluation (PE) method to interpret the performance of ARF-Reg model on the stream. In this method, each newcomer samples are used to test (prediction) before train the model [27]. It means all data samples are used to either test or train the model. PE has following parameters.

- **n_wait** refers to the number of samples to be processed between each estimate.
- **max_samples** means the number of all samples to be processed during the evaluation process.
- **batch_size** indicates the number of samples that will enter the model at the same time.
- **pretrain_size** means the number of samples used to train the model before any prediction takes place.
- **metrics** refers to the metrics used to measure model performance.
- **output_file** used to write the detailed summary of evaluation to a file.

- **show_plot** shows the “current” and “mean” values of evaluation metrics as a dynamic plot.

We evaluated the performance of ARF-Reg with some measures such as Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE). In addition, we have also showed values such as training time, test time and used memory of ARF-Reg for each data set. The dynamic plots that we will present during the evaluation show the performance of the ARF-Reg model which we have established on the stream. Prequential Evaluation method does not calculate the RMSE value, we cannot see it in the dynamic plots. Thus, we manually calculated the RMSE values for each data set.

3.4.1. Performance Evaluations

As mentioned section 3.4.3., hyperparameter tunning is very difficult and complex process in streaming learning and there is not any exact solution to this challenge in literature. So, we tried multiple combinations manually to find optimum parameter of ARF-Reg and we obtained the best results when *m* (feature size) = 4, *n* (number of trees) = 50, and ADWIN as drift detection method.

We set parameters of PE like in Table 3.4. to evaluate performance of ARF-Reg model for the FD001 dataset.

Table 3.4. Parameter values of Prequential Evaluation for FD001 dataset.

max_samples	33727
pretrain_size	200
show_plot	True
n_wait	1
metrics	['mean_square_error','mean_absolute_error', 'running_time','model_size','true_vs_predicted']
output_file	'filename.csv'

We can see the performance evaluation of the model on the data from the stream as a dynamic plot. Model performance has been evaluated in two ways (*Mean* and *Current*) at various points on the dynamic plot. While *mean* means average performance on the data seen so far, *current* is the performance on the most recent data. We have also measured resources such as time and memory. Figure 3.4. indicates performance evaluations of ARF-Reg model on the FD001 dataset.

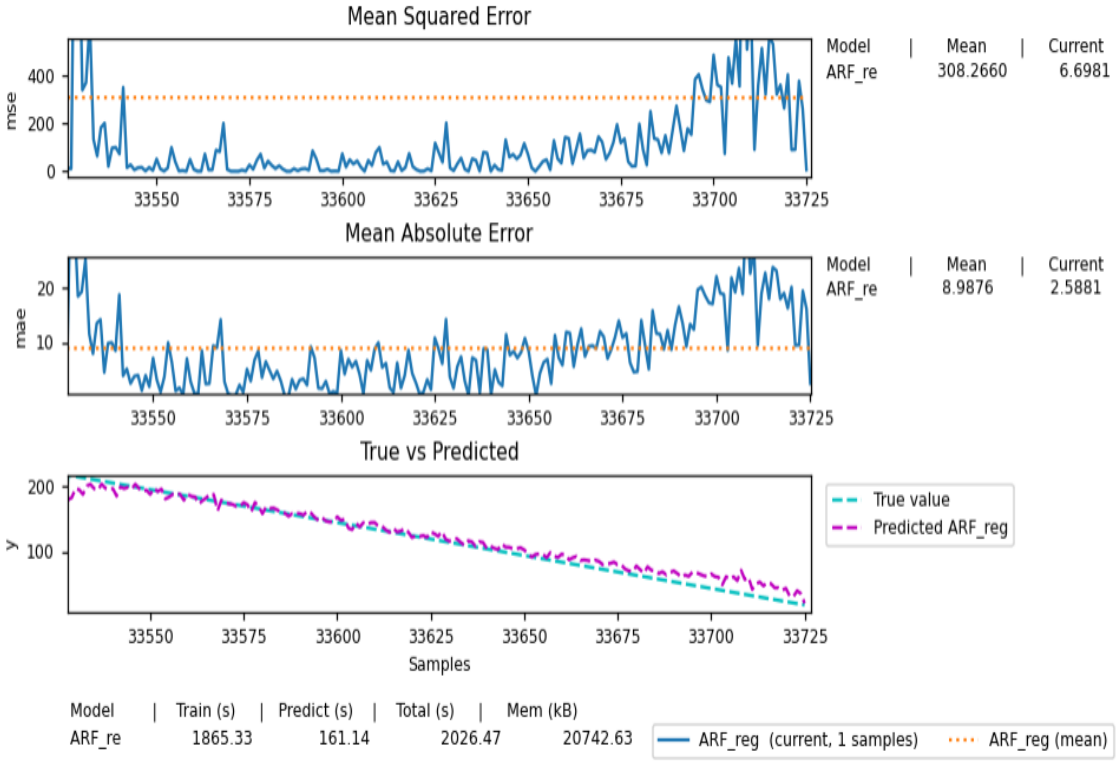

Figure 3.4. Performance evaluation of ARF-Reg on the FD001 dataset.

We can realize performance of ARF-Reg which MSE is 308.2660 and MAE equals to 8.9876. Moreover, we have calculated RMSE for ARF-Reg and it equals to 17.5517. It is important to point out that *running_time* and *model_size* is shown with their only *current* values on the dynamic plot. As a result, in the FD001 dataset, the total train time of the samples was 1865.33 seconds, and the test time was 161.14 seconds.

Table 3.5. shows parameter settings of PE for FD002 dataset in detail.

Table 3.5. Parameter values of Prequential Evaluation for FD002 dataset.

max_samples	87750
pretrain_size	1000
show_plot	True
n_wait	1
metrics	['mean_square_error','mean_absolute_error', 'running_time','model_size','true_vs_predicted']
output_file	'filename1.csv'

We also used the same parameter settings of ARF-Reg which is mentioned at the beginning of this section. Figure 3.5. illustrate performance evaluations for ARF-Reg model with several metrics.

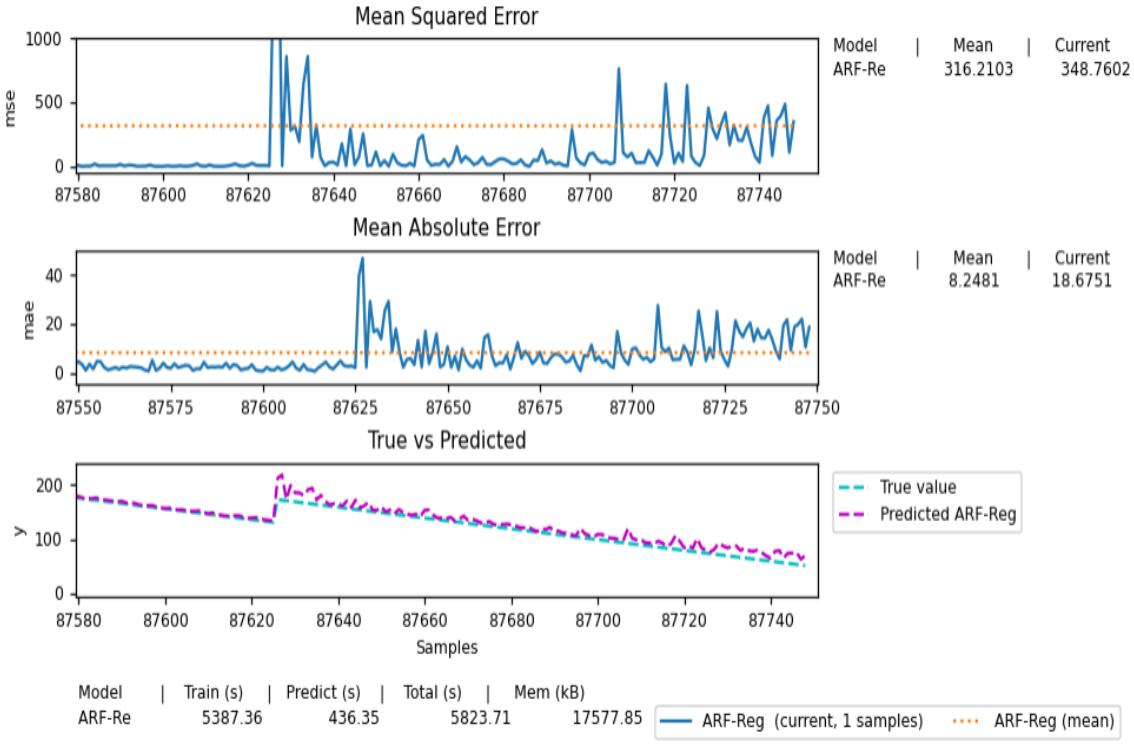

Figure 3.5. Performance evaluation of ARF-Reg on the FD002 dataset.

Performance evaluation of ARF-Reg on the FD002 dataset is like that: MSE is 316.2103, MAE is 8.2481 and the extra calculated RMSE value is 17.7823. The total train time of the samples was 5387.36 seconds, and the test time was 436.35 seconds. For FD003 dataset, parameters values of PE are set like in Table 3.5. but we change *max_samples* value to 41315 because FD003 dataset includes 41315 samples and then we evaluate performance of ARF-Reg for this dataset.

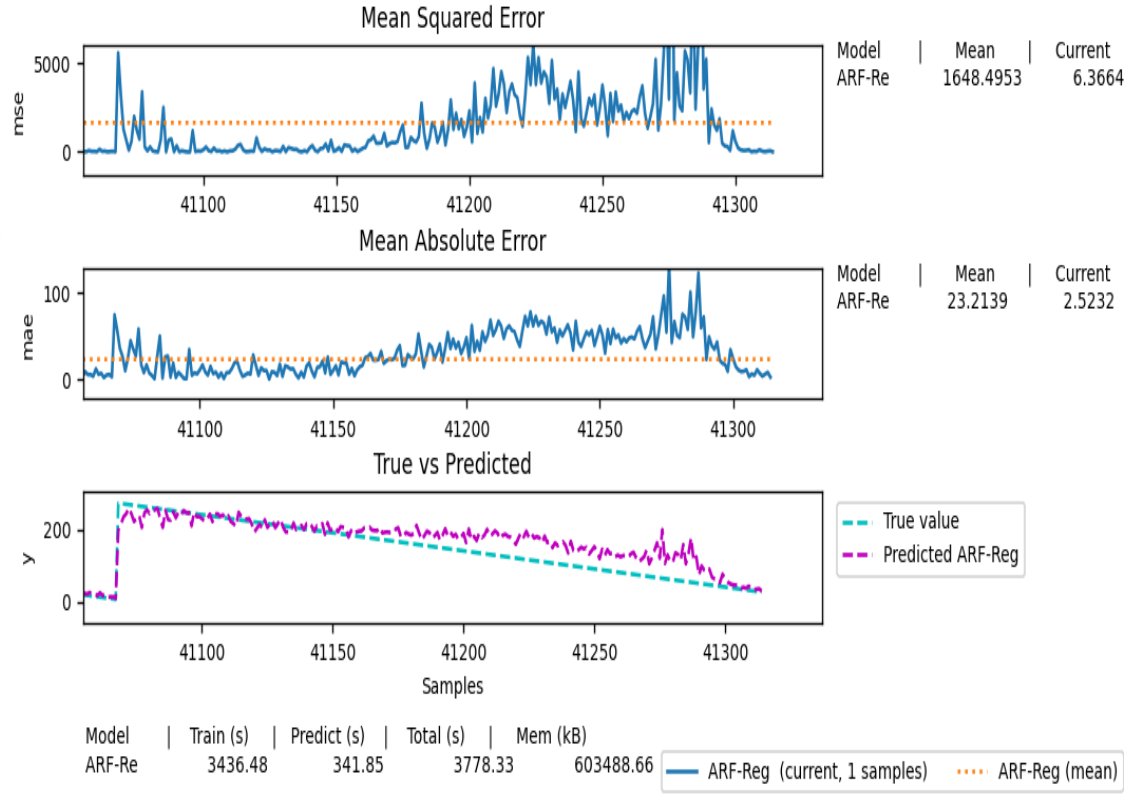

Figure 3.6. Performance evaluation of ARF-Reg on the FD003 dataset.

According to Figure 3.6. MSE is 1648.4953, MAE is 23.2139 and calculated RMSE is 40.6016. The total train and test time of the samples were 3436.48 second, and 341.85 second, respectively.

Following results belong to performance evaluation of ARF-Reg on the FD004 dataset. The same parameter values of PE are set for this dataset and again we change *max_samples* value to 102463 because of number of samples in FD004 dataset. Figure 3.7. depicts detailed information about performance evaluation of ARF-Reg.

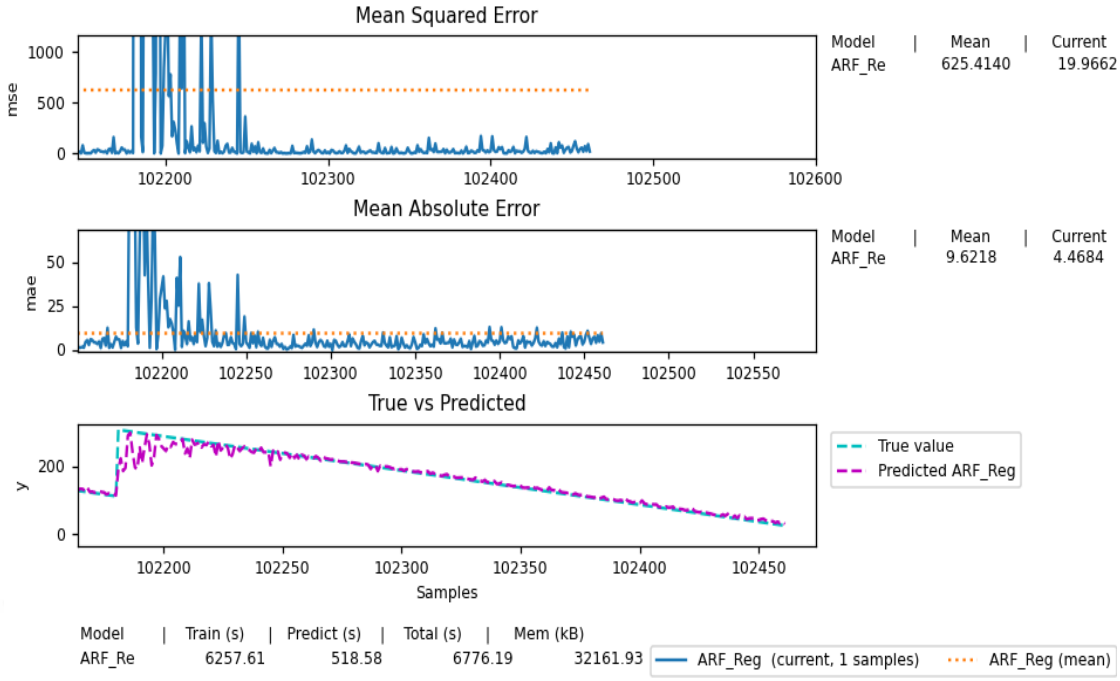

Figure 3.7. Performance evaluation of ARF-Reg on the FD004 dataset.

As we can see on right side of Figure 3.7., MSE and MAE values are 625.4140 and 9.6218, respectively. Also calculated RMSE value is 25.0082. The total train and test time of the samples was 6257.61 second, and 518.58 second, respectively. In all datasets, each sample was trained approximately at between 0.04 – 0.08 second and tested at between 0.004 – 0.008second.

Finally, we have shown the performance evaluation of the ARF-Reg model in a single table for the four data sets which we used. Table 3.6. illustrate MSE, MAE and RMSE values of ARF-Reg for each four datasets.

Table 3.6. MSE, MAE and RMSE values of ARF-Reg for each four datasets

Dataset	**MSE**	**MAE**	**RMSE**
FD001	308.2660	8.9876	17.5517
FD002	316.2103	8.2481	17.7823
FD003	1648.4953	23.2139	40.6016
FD004	625.4140	9.6218	25.0082

It is important to note that above results based on whole dataset. It means, we combine train and test sets, so they were handled together because we used all samples for test then train. Whereas we also calculated RMSE values of ARF-Reg on test sets for comparison with related studies which are mentioned chapter 1. Table 3.7. depicts RMSE values of ARF-Reg on test sets.

Table 3.7. RMSE values of ARF-Reg on test sets.

Datasets	**RMSE**
FD001	18.1560
FD002	18.4601
FD003	48.4478
FD004	27.9401

In addition, we mentioned above that the detailed summary of the performance evaluation results is written in a file. In this file we can also see the actual values of RUL which show how many cycles the engines will fail, and our predicted RUL values.

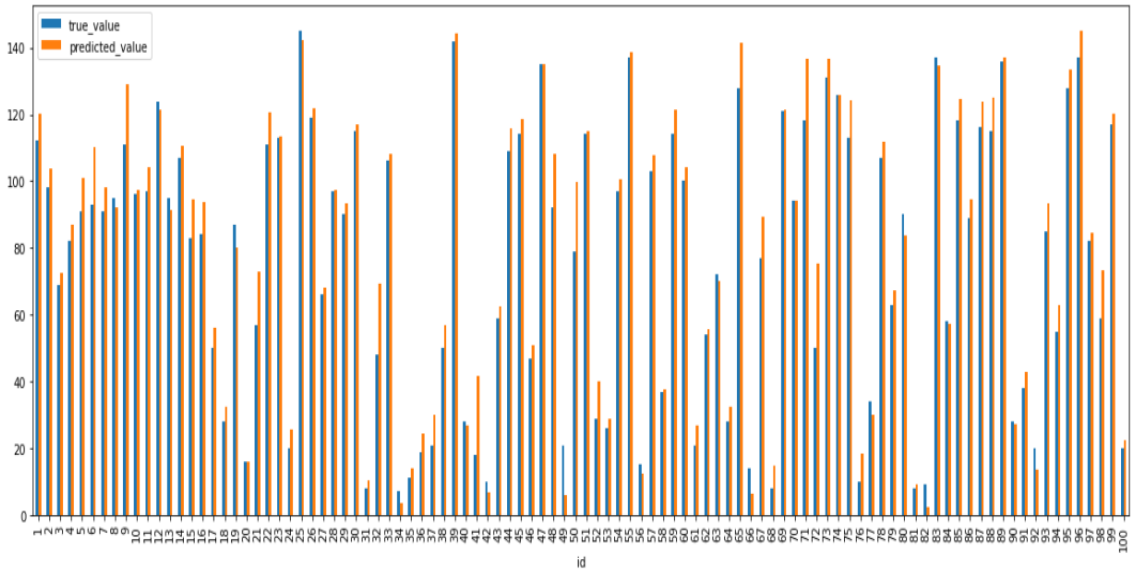

Figure 3.8. Predicted and actual values of RUL for FD001 dataset.

Based on test set of FD001 dataset Figure 3.8. indicates comparison between actual values of RUL and our predicted values using ARF-Reg. x axis and y axis of the Figure 3.8. shows ID number and RUL cycles of each engines, respectively. For instance, according to actual value, engine 1 will fail after 112 cycles, our model predicted it as 120 cycles.

In the same way, according to Figure 3.8. engine 2 will fail after 98 cycles, prediction of ARF-Reg is 103 cycles.

Also, we can see comparison between actual values of RUL and our predicted values using ARF-Reg for FD003 dataset on Figure 3.9. Considering the difference between the predicted and actual values in this figure and Figure 3.8., we can easily see that the performance of ARF-Reg on the FD001 data set is better than the performance on the FD003 data set.

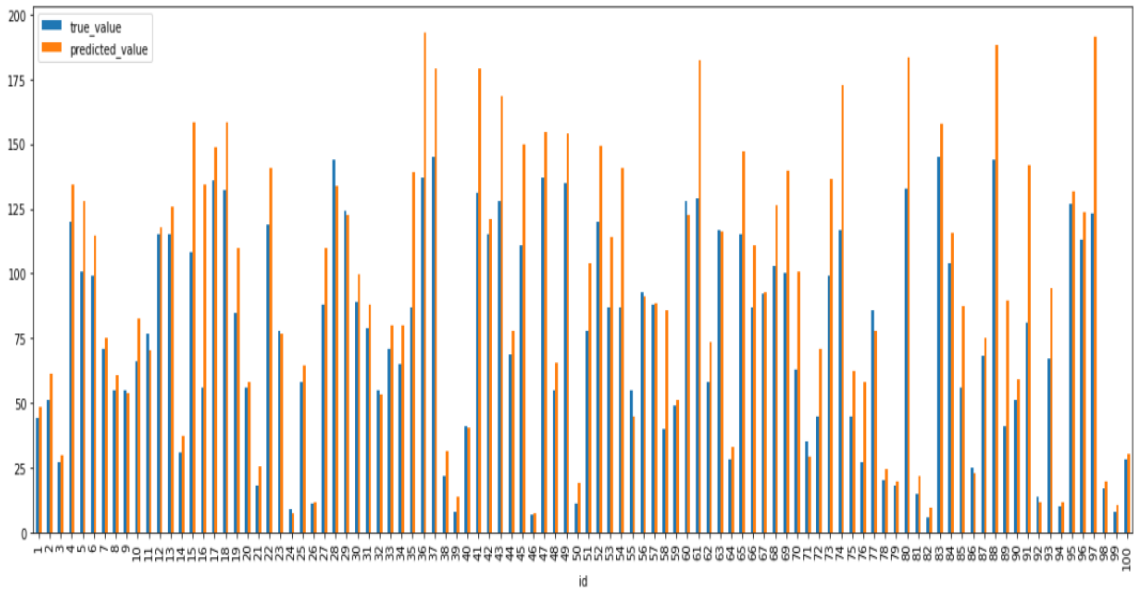

Figure 3.9. Predicted and actual values of RUL for FD003 dataset.

As seen from this Figure 3.9., while the actual RUL cycle of the first engine is 44, our ARF-Reg model estimated it as 48. In another point, actual RUL cycles of engine96 is 113 but prediction of our model is 123 cycles.

In addition, scikit-multiflow provides some advantages. One of them is that it is able to evaluate multiple models at the same time. As an example, to this, we compared the performance of ARF-Reg which we used in our study and HAT models. Figure 3.10. illustrates performance evaluations of both methods. MSE and MAE values are represented on the right side of the dynamic plot. Also, we can see time (train and testing) and memory values for each method on the bottom of the plot.

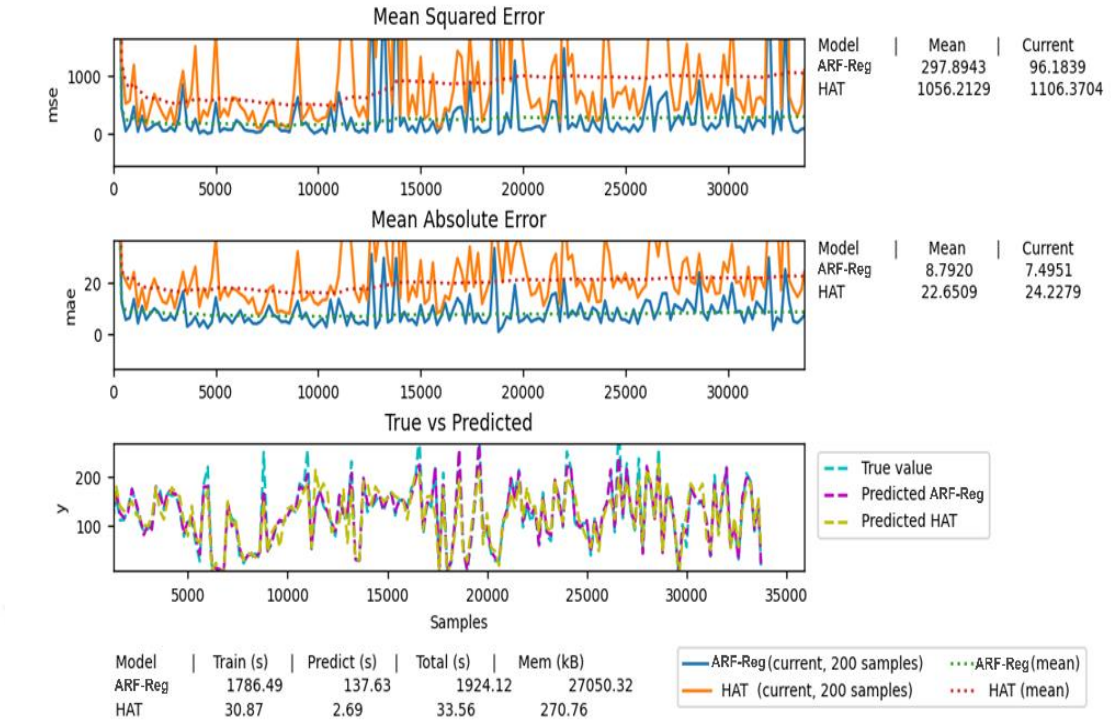

Figure 3.10. Performance evaluations of ARF-Reg and HAT methods.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1. Conclusion

In this study, regression method with ARF-Reg which is one of the most popular stream learning methods is proposed in order to predict Remaining Useful Lifetime of engines. Four Turbofan Engine Degradation Simulation Datasets which were taken from Nasa data repository were used in this study. The main objective of the study is to use stream learning method (Adaptive Random Forests for Regression) on mentioned datasets and talk about advantages of stream learning against to conventional batch learning.

Firstly, we combined train and test sets to handle them together in each four datasets because of nature of streaming learning and then data prepare step was applied to be suitable for ML model. Data preprocessing operations such as feature selection and data transformation were handled as offline because currently there is not any exact solution to such operations for streaming learning. So, Min-Max transformation method was applied to rescale feature into [0-1] range and Pearson's Correlation Coefficient method was used select the most relevant feature, respectively.

Prequential Evaluation method where each sample are used to test then train, was applied to evaluate performance of ARF-Reg on each four datasets. We evaluated performance of ARF-Reg using Mean Squared Error, Mean Absolute Error which are metrics of Prequential Evaluation. We also calculated Root Mean Squared Error values of ARF-Reg for each dataset.

As mentioned previously, results of performance evaluation were obtained from whole dataset due to combine train and test sets. In addition, we also calculated Root Mean Squared Error values of ARF-Reg on only test set to compare with other batch learning based studies in which model performances were evaluated on test set. Results indicated that ARF-Reg was not bad at all compared with other studies which handled the same datasets in overall performance. In addition, stream learning is more effective approach compared with batch learning in dynamic environment because stream learning

models update themselves according to new incoming data and can cope with concept drifts. In addition, in systems based on batch learning, as data increases over time, investments in resources such as memory and training time also increase. However, in stream learning systems, resources are managed more effectively, which makes stream learning an advantageous approach for big data applications.

4.2. Future Works

There are various data preprocessing methods for batch learning in the literature, but it is very difficult to say the same for stream learning. Due to the nature of stream learning, it is necessary to perform the data preprocessing steps as online and systematic with streamed data, that is, in a way that can be applied to the streaming data. In addition, hyperparameter tunning of streaming models must be adapted to the streaming data. There are not any exact solutions for these issues in literature today. So, it is very difficult to apply stream learning methods for real-world application in efficient way.

Future studies may include to find solutions for such challenges to apply stream learning techniques to real world problems and evaluate the performance of models more realistically. Also, several stream learning methods may be applied on different kind of datasets and compared each other.

REFERENCES

[1] Mathew, V., Toby, T., Singh, V., Rao, B. M., & Kumar, M. G. (2017, December). Prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning. In 2017 IEEE International Conference on Circuits and Systems (ICCS) (pp. 306-311). IEEE.

[2] Bruneo, D., & De Vita, F. (2019, June). On the use of LSTM networks for Predictive Maintenance in Smart Industries. In 2019 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 241-248). IEEE.

[3] Babu, G. S., Zhao, P., & Li, X. L. (2016, April). Deep convolutional neural network based regression approach for estimation of remaining useful life. In International conference on database systems for advanced applications (pp. 214-228). Springer, Cham

[4] Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017, June). Long short-term memory network for remaining useful life estimation. In 2017 IEEE international conference on prognostics and health management (ICPHM) (pp. 88-95). IEEE,

[5] Fornlöf, V. (2016). Improved remaining useful life estimations for on-condition parts in aircraft engines (Doctoral dissertation, University of Skövde).

[6] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2014). Machine learning for predictive maintenance: A multiple classifier approach. IEEE Transactions on Industrial Informatics, 11(3), 812-820.

[7] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 137, 106024.

[8] Baştanlar, Y., & Özuysal, M. (2014). Introduction to machine learning. miRNomics: microRNA biology and computational analysis.

[9] Montiel, J. (2020). Learning from evolving data streams. Proceedings of the 19th Python in Science Conference.

[10] Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2011). Data stream mining a practical approach. Journal of Machine Learning Research, 11 (2010) 1601-1604.

[11] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM computing surveys (CSUR), 46(4), 1-37.

[12] Domingos, P., & Hulten, G. (2000, August). Mining high-speed data streams. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 71-80).

[13] Bifet, A., & Gavaldà, R. (2009, August). Adaptive learning from evolving data streams. In International Symposium on Intelligent Data Analysis (pp. 249-260). Springer, Berlin, Heidelberg.

[14] Bifet, A., & Gavalda, R. (2007, April). Learning from time-changing data with adaptive windowing. In Proceedings of the 2007 SIAM international conference on data mining (pp. 443-448). Society for Industrial and Applied Mathematics.

[15] Montiel, J. (2020). Learning from evolving data streams. Proceedings of the 19th Python in Science Conference.

[16] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[17] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

[18] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., ... & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. Machine Learning, 106(9-10), 1469-1495.

[19] Domingos, P., & Hulten, G. (2000, August). Mining high-speed data streams. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 71-80).

[20] Gomes, H. M., Read, J., Bifet, A., Barddal, J. P., & Gama, J. (2019). Machine learning for streaming data: state of the art, challenges, and opportunities. ACM SIGKDD Explorations Newsletter, 21(2), 6-22.

[21] AlNuaimi, N., Masud, M. M., Serhani, M. A., & Zaki, N. (2019). Streaming feature selection algorithms for big data: A survey. Applied Computing and Informatics.

[22] Montiel López, J. (2019). Fast and slow machine learning (Doctoral dissertation, Université Paris-Saclay (ComUE)).

[23] Gama, J., Sebastião, R., & Rodrigues, P. P. (2009, June). Issues in evaluation of stream learning algorithms. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 329-338).

[24] Frederick, D. K., DeCastro, J. A., & Litt, J. S. (2007). User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS).

[25] Aydin, O., & Guldamlasioglu, S. (2017, April). Using LSTM networks to predict engine condition on large scale data processing framework. In 2017 4th International Conference on Electrical and Electronic Engineering (ICEEE) (pp. 281-285). IEEE.

[26] Qi, L., Zhanbao, G., Diyin, T., & Baoan, L. (2016). Remaining useful life estimation for deteriorating.

[27] Dawid, A. P. (1984). Present position and potential developments: Some personal views statistical theory the prequential approach. Journal of the Royal Statistical Society: Series A (General), 147(2), 278-290.

[28] Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. The Journal of Machine Learning Research, 19(1), 2915-2914.

[29] Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. Neurocomputing, 239, 39-57.

[30] Gomes, H. M., Barddal, J. P., Ferreira, L. E. B., & Bifet, A. (2018, April). Adaptive random forests for data stream regression. In ESANN.

[31] Kulkarni, K., Devi, U., Sirighee, A., Hazra, J., & Rao, P. (2018, June). Predictive maintenance for supermarket refrigeration systems using only case temperature data. In 2018 Annual American Control Conference (ACC) (pp. 4640-4645). IEEE.

[32] Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., & Loncarski, J. (2018, July). Machine learning approach for predictive maintenance in industry 4.0. In 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA) (pp. 1-6). IEEE.

[33] Biswal, S., & Sabareesh, G. R. (2015, May). Design and development of a wind turbine test rig for condition monitoring studies. In 2015 International Conference on Industrial Instrumentation and Control (ICIC) (pp. 891-896). IEEE.

[34] Kanawaday, A., & Sane, A. (2017, November). Machine learning for predictive maintenance of industrial machines using IoT sensor data. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 87-90). IEEE.

[35] Dong, D., Li, X. Y., & Sun, F. Q. (2017, July). Life prediction of jet engines based on lstm-recurrent neural networks. In 2017 Prognostics and System Health Management Conference (PHM-Harbin) (pp. 1-6). IEEE.