Optimising maintenance schedules for a logistic fleet with predictive maintenance

Cedric Plouvier*, Valentin Carlan† and Peter Hellinckx*
*Faculty of Applied Engineering - Department of Electronics-ICT
University of Antwerp, Antwerp, Belgium
†Digitrans
Antwerp, Belgium

***Abstract*—The logistics sector is a fast growing market thanks to globalisation driven by numerous technological innovations. Just like any market it is important to gain an advantage over your competitors. In this paper we investigate whether predictive maintenance instead of traditional reactive and preventive maintenance can create this advantage. With the rapid increase and development of computing power AI is a promising technology to predict when maintenance is needed. Other works and use cases from different industries have already shown promising results for predictive maintenance with AI. This research focuses on predicting the need of tire replacement since this is an often recurring and high maintenance cost. In the first part of the paper we focus on gathering the data from the trucks. After that we train, optimise and evaluate different regression models to predict the amount of weeks before a tire repair is needed. This is done with an auto machine learning tool called TPOT which will find the optimal pipeline for us after which we further optimise the parameters. The results show the technical feasibility and potential of predictive maintenance for tire maintenance but that more data of higher quality is needed to become an economically viable predictive maintenance schedule.**

I. Introduction

Artificial Intelligence (AI) has been a trigger for innovation over the past decades. From the first chat bot named Eliza, created by Joseph Weizenbaum at MIT in 1966 [1], to a more recent milestone as AlphaGo in 2016 [2]. Thanks to rapid developing core technologies such as computer power, big data, cloud computing and Internet of Things connectivity, AI is able to prove its full potential.

The logistics sector is a large and price competitive industry in which companies adopting AI can gain a competitive advantage. An even more important factor for logistic companies to keep investing in innovation and adoption of AI is the risk of becoming obsolete in the future. Another reason why the logistics sector is suited for AI adoption is the high amount of data available. Nevertheless, according to a study from IBM and DHL in 2018, only 10% of the logistics industry uses AI analysis and results [3].

The most popular AI application in the logistic sector is autonomous driving but many more are developed and used. IBM and DHL see use cases for AI in the back office to remove repetitive data processing tasks using Natural Language Processing (NLP), detect financial anomalies such as fraudulent invoices, keep costumer information up to date, automate customs brokerage processes. They also develop AI use cases for predictive purposes such as shipment delays, demand forecasting, route optimization. Also sorting and visual inspections can be improved with AI systems [3].

In this research we focus on predictive maintenance (PdM) for a truck fleet using AI. Traditional ways of maintenance are reactive maintenance and preventive maintenance. Reactive maintenance only repairs a vehicle when broken. This results in large costs and high downtime when parts break down. Preventive maintenance counters these downsides of reactive maintenance by implementing a maintenance schedule resulting in low repair cost but often unnecessary downtime and cost for inspection. Predictive maintenance looks for the optimal between reactive and preventive maintenance by replacing parts before they break down while avoiding unnecessary scheduled maintenance. Predictive maintenance can be implemented without AI. Often the remaining useful lifetime (RUL) is calculated based on mathematical models and their distribution for different parts of a vehicle. We believe this method is sub optimal compared to AI methods. Models quickly become too complicated or simplify reality. Factors such as weather or truck load are often ignored in mathematical models. AI permits us to find correlations and predict the RUL taking these complicated factors into account. AI can not only be used to predict RUL but also detection of anomalies. Abnormal behaviour of components or amount of log messages get detected and signals the need of maintenance. The focus of the paper is to predict the RUL of tires on trucks using AI with basic data logistics companies have at their disposal. Results show that with basic and minimal data predictions can be made for tire wear. The results can improve further if more complex data is added. In Section II we discuss related works to our research. Section III elaborates on the methods and results. These results are then discussed in section IV after which we discuss these on section V. In Section VI we come to a final conclusion of the study.

II. Related work

In this section we gather relevant information on PdM. We discuss papers using AI techniques but also papers with statistical approaches. We do not only discuss PdM in the logistics sector since other sectors and general overviews give

valuable insights. A extensive overview is given in *A survey of predictive maintenance: Systems, purposes and approaches* [4]. Three ways of approaching PdM are discussed. Knowledge based, Traditional machine learning based and deep learning based. Knowledge based approach can be classified into ontology-based, rule-based and model based. The survey states that knowledge based approaches needs many assumptions and are not resilient to changes. Examples of Traditional machine learning approaches are Artificial Neural Networks (ANN), Decision Tree (DT), k-nearest Neighbors (k-NN) and Support Vector Machines (SVM). Discussed deep learning based approaches are Auto-Encoders (AE), Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN), Deep Belief Networks (DBN), Generative Adversial Network (GAN), Transfer Learning, Deep reinforcement Learning (DRL). Tables with advantages, limitations and typical applications can be used as guidelines. *Wang et al.* [5] estimates RUL based on a knowledge model based approach for helical springs, brake pads, tires and vehicle engines by using a multi-objective evolutionary algorithm. Another non-AI approach to PdM is the Consesus self-organised models for fault detection (COSMO) [6]. It is a fault detection strategy to find faults that or not predefined. Where interesting features are detected on board of vehicles or other objects using normalized mean square error between signals. A central server then detects deviations based on a Gaussian distribution fit. This research is followed with a long-time case study from *Byttner et al.* that reduced downtime of a fleet of city buses significantly. Feature selection on board is done with auto encoders [7]. *Killeen et al.* improves the sensor selection performed in COSMO with semi-supervised machine learning [8]. Also *Holst et al.* [9] implements a PdM technique based on anomalies is implemented for a train fleet. Based on Bayesian statistics a model of on board logged event messages get build and new samples are compared to the model to determine if it is normal or an anomaly. In *Toward Supervised Anomaly Detection* further research show that changing anomaly detection from a unsupervised task to a semi-supervised task where edge cases get labeled by experts is beneficial to improve accuracy. *Umeda et al.* [10] proposes a maintenance schedule update method for plasma Etchers. Based on sensor data the RUL is calculated using histograms of past failures with similar trend. When the risk of failure cost becomes high the additional maintenance is included in the next scheduled maintenance. *Wang et al.* [11] researches predictive maintenance for high-speed railway trains with long short-term memory (LSTM) networks. Failure samples are generated by the failure model of high-speed railway power equipment. Together with normal sensor data a RUL is obtained by the LSTM deep learning model.

III. Methodology

First step of the research is to assess the available data. Then data is be processed into a database. Users interact with the data through a web-based application build on the java spring boot and thymeleaf framework. Based on the available data we determine which AI techniques are most suited to implement a PdM schedule. Data is then prepared for training which results in a model used to predict the next maintenance date.

A. Data

A logistics company with a truck fleet of 250 trucks and 457 trailers has the following data available.

1) Trucks: It is not an homogeneous fleet. The trucks are from different brands: Renault, Mercedes, Daf, Volvo, Man and Scania. A third party tool called Transics is used to receive remote diagnostics. These diagnostics are real-time and are also later accessible through the Transics API [12]. The API allows 20 requests per minute. From the Transics dashboard log reports can be exported in excel format. We can extract following reports from Transics: Activities, distances, consumption, load. Except for the activities all data is accumulated over an entire day. This limits the granularity of the data and the possibility to couple info of load and consumption to each activity. Transics data is available from 2016 until 2020

2) Trailers: The trailer fleet exists of multiple types like platform trailers, fridge, tippers, walking floor trailers and containers. All trailers are equipped with a low power sensor of Sensolus. The trackers have movement and tilt detection. Three states are indicated by the movement detection with corresponding GPS locations: start, on the move, stop. Last known position is visible on the Sensolus web platform where use can generate log reports. Sensolus API accepts 5000 calls every month. Webhooks or push API update users when the state of movement changes. Webhooks do not count towards the API limit.

3) Maintenance: Maintenance of trucks and trailers are always done at the same garage since 2010. For every maintenance a PDF template invoice is created describing what repairs are done together with the price. The logistic company create a monthly overview of total cost for every truck and trailer and list of corresponding invoice numbers. Maintenance invoices are available from 2010 until 2020.

B. Business logic spring application

With the spring boot framework we persist data into a h2 in-memory database. The spring framework is developed according to the model-view-controller design principle (MVC) with Spring MVC. The following sections elaborate on the functionality of the web application.

1) Fleet management: The fleet of trucks and trailers can be managed through the web interface. Users can add or remove vehicles. Parameters such as number plates and chassis number can be modified. Vehicle database identifiers are not based on number plate or chassis number. This way the collected data is always correctly linked to the actual vehicle in case number plates get interchanged. New data excels and invoices are added through the web application. Truck and Trailer parameters such as number plate and chassis number can be changed.

2) Maintenance management: Import new invoices through the web application. Invoices are then processed with a PDF parser to persist data into the database. Persisted data are invoice number, data, client number, vehicle and maintenance type coupled to the corresponding cost. Only most common maintenance types are detected while undefined maintenance types who are uncommon are classified as unclassified. The PDF parsing algorithm sets a Boolean flag if the invoice contains a maintenance type that is of importance for out PdM such as tire repairs in our case. This boolean is used to obtain the dates between 2 tire repairs. Once the PDF's are parsed and persisted into the database users can view the classified cost of a list of vehicles over a period of time.

3) Couple Trailer to Truck: The use of different third party applications result in limited insight which trailer a truck uses for a task. The functionality is possible in Transics but reality shows this system does not work well. Input is often wrong or incomplete and paper trip sheets are still used. Therefor we implemented an algorithm independent of human interaction to couple trucks with trailers. This way we obtain better insights in client profitability. Thanks to the algorithm we can also extrapolate Transics data about load or consumption onto the trailers. The pseudo code of the algorithm is given underneath.

for state messages **do**
  if state equals START **then**
    transicsTruckActivity([t-15;t+15])
    $previousDistance = 20000$
    if $distanceToTrailer \leq previousDistance$ **then**
      $possiblebleTrucks \leftarrow < truck; distance >$
    end if
  end if
  if state equals STOP **then**
    transicsTruckActivity([t-15;t+15], $possibleTrucks$)
    calculate closest average truck start and stop
    assign truck to trailer activity
    assign travelled distance to trailer activty
    assign average speed to trailer activity
  end if
end for=0

C. Predictive Maintenance

1) Component choice: The available truck data from Transics is not specific to a component and is quite generic. Components we assume that wear out due to driving distance, driving speed, load and motor time are lights, tires, break pads, suspension and the motor itself. Since only 4 years of data are available we take into account that components that do not break frequently to obtain enough data points to train a model. By manually going through the maintenance invoices we observe that break pads replacements, suspension and motor repairs are not that frequent and only give us a limited amount of data points to train a model. Light repairs are common but only correlate with the time trucks driving and not other available data such as speed and load. We decide that tires are the most interesting candidate since they correlate with all available data and sufficient tire repairs are executed over the past 4 years.

2) Model choice: The result or target we want from our model is time until the next tire repair. Since we have the time between two tire repairs of a truck we can choose for supervised learning models. Examples of supervised learning algorithms are Naive Bayes, regression, SVM, k-NN, random forest and neural networks. AI models take long to train with multiple hyper parameters to configure resulting in a long process of trial and error to find a suited model. Therefor we use TPOT which is an auto machine learning tool (ML) build on the scikit-learn python machine learning library. TPOT uses genetic programming (GP) which is a technique of evolving programs to fit a particular task. TPOT investigates pipeline configurations and results in the most suited pipeline for our data. Nevertheless TPOT takes a long time to obtain good results and different runs result in different pipelines. TPOT solves classification and regressions problems. Neural networks are only implemented for classification problems but future releases for regressions problems will be released. For our PdM use case regression is most suited.

3) Data preparation: All data is in our h2 database. Next step is to calculate the cumulative data between two tire repairs for a certain truck. Figure 1 shows the cumulative data class in our database.

This data need to transform into meaningful data for the regressions model. TPOT needs the regression data in a CSV format with the data split into features and target data which are the weeks between repairs. Table 1 shows the input CSV for TPOT regression training. In total there are 1251 tire repairs for the trucks between 2016 and 2020.

Type	Data
kmTravelled	feature
kmCounterStart	feature
kmCounterEnd	feature
motorTime	feature
drivingtime	feature
drivingPercentage	feature
StationaryTime	feature
StationaryPercentage	feature
averageSpeed	feature
soloPercentage	feature
soloKm	feature
emptyPercentage	feature
emptyKm	feature
unknownPercentage	feature
unknownKm	feature
loadKm	feature
loadPercentage	feature
weeks	target

TABLE I
CSV FILE FEATURES AND TARGET.

After creating the CSV file we manually investigate it. Some data is faulty and after debugging the origin of the faulty data is not the accumulation algorithm but the Transics API. We manually remove following unrealistic data from the CSV file: average speeds above 130, abnormal high travelled kilometers

CumulativeDataBetweenTireRepairsTrucks	
id	**int**
truck	Truck
numberplate	string
datePreviousRepair	date
dateCurrentRepair	date
startKilometerCounter	int
endKilometerCounter	int
motorTimeHours	duration
drivingTimeHours	duration
drivingTimePercentage	double
stationaryTimeHours	duration
stationaryTimePercentage	oduble
averageSpeedDriving	double
soloPercentageDriving	double
soloKilometersDriving	int
emptyPercentageDriving	double
emptyKilometersDriving	int
unknownPercentage	double
unknownKilometers	int
loadedKilometers	int
loadedPercentage	double
weeksBetween	int

Fig. 1. Features of the ML model

that don't match the kilometer counter, remove negative values for unknown kilometers and unknown percentages.

Next we analyse the relation between our features and target. A correlation matrix is calculated and is shown in Figure 2.

Analysis of the correlation matrix show logic correlations. High correlation between motor time, driving time and kilometers travelled. Our target, the amount of weeks, correlates high with kilometers travelled. We also remark a correlation for kilometers travelled with kilometers driving empty and kilometers driving loaded. This is simply because the more kilometers travelled one or the other has to increase with it.

4) TPOT: Libraries of TPOT can run in any python environment. We used PyCharm by Jetbrains. The following steps are automated by TPOT:

1) Feature selection, preprocessing and construction
2) Model selection
3) Parameter optimization

A regression template provided by TPOT on the website is the

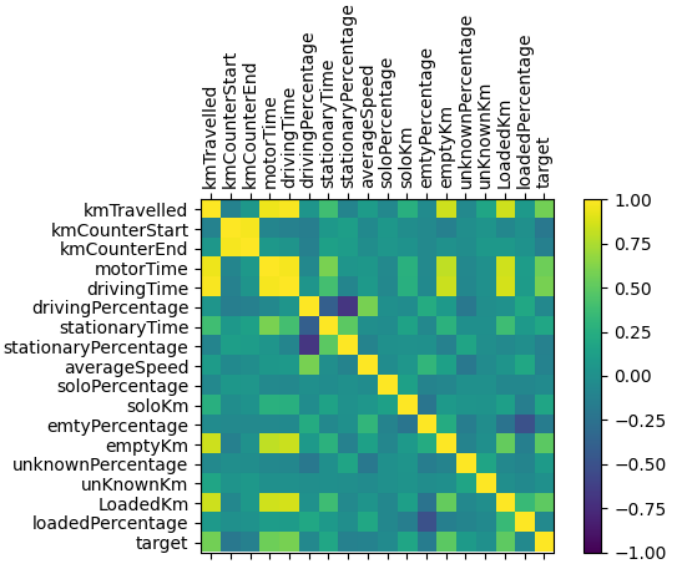

Fig. 2. Correlation Matrix of features data

starting point to find the best pipeline [13]. It is best practice to split our data into a training set and a testing set. That way we make sure our model is not trained to fit our test or validation set. TPOT suggests to use 75% of data for training and 25% for testing. Resulting in 938 data points for training and 313 for testing. Too many training samples cause overfitting while insufficient training samples cause underfitting. The TPOT regressor arguments are the most important to find the optimal pipeline. The amount of generations is the number of iterations to run the pipeline optimization process. Every generation has an optimal pipeline as outcome. The population is the amount of individuals to retain in every generations. A TPOT regressor with hundred generations and hundred populations will result in the evaluation of ten thousand pipelines. The random number parameter assures that outcomes are reproducible for a same data set. This is important to compare different pipelines when testing different regressor parameters. Multiple pipelines are generated with TPOT after which we test and tune the hyperparameters.

IV. Results

The first step in discussing results is defining how we measure the results. Standard TPOT scoring function is R squared. The target of the model is to predict the weeks before tires need repair. The error of the model on the testing set is calculated with the Root Mean Squared Error (RMSE). The interpretation of the RMSE is the average amount of weeks the model predicted wrong on the testing set. The R squared scoring function and the RMSE are different scoring functions that we compare. The benefit of RMSE is that we can interpret it against our average amount of weeks between tire repairs while R squared gives us a more abstract score. This RMSE needs to be compared to the average weeks between tire repair to gain insights if the model is sufficiently accurate.

The average of weeks between tire repairs between 2016 and 2020 is 29.6 weeks.

A. pipelines

1) XGBRegressor 1: The first pipeline we obtain is based on the data without removing the unrealistic data discussed in section data preparation. For the first pipeline search we use the TPOT standard recommendations of hundred generations and population size of one hundred. The used parameters are as follows:

- Generations: 100
- Population size: 100
- Random state: 42
- Training size: 75%

Exported pipeline: XGBRegressor(learning rate=0.1, max depth=8, min child weight=3, n estimators=100, n jobs=1, objective="reg:squarederror", subsample=0.8500000000000001, verbosity=0)

XGB stands for extreme gradient boosting. Next we evaluate the pipeline based on the Cross Validation score on the training set but more important the RMSE of the predictions on our testing data. We always fit the exported pipeline on a data training size of 75%, 85% as well as 65% to detect possible over- or underfitting. This respectively corresponds to testing sizes of 25%, 15% and 35%.

- Average CV score: -548
- RMSE with 75% training and 25% testing data: 25.13
- RMSE with 85% training and 15% testing data: 25.04

2) XGBRegressor 2: For the second pipeline the unrealistic data was removed. The standard suggested generations and population size did not change.

Pipeline search parameters:

- Generations: 100
- Population size: 100
- Random state: 42
- Training size: 75%

Exported pipeline: XGBRegressor(learning rate=0.1, max depth=8, min child weight=3, n estimators=100, n jobs=1, objective="reg:squarederror", subsample=0.8500000000000001, verbosity=0)

Pipeline results:

- Average CV score: -548
- TPOT score function: 0.99920
- RMSE with 75% training and 25% testing data: 16.67
- RMSE with 85% training and 15% testing data: 17.23
- RMSE with 65% training and 35% testing data: 22.92

Removing the data has a positive influence on the prediction score. This underlines the importance of data quality when training ML models.

3) Gradient Boosting Regressor: When running a new pipeline search with the same parameters and random state we expect to obtain the same results as the previous search.

Pipeline search parameters:

- Generations: 100
- Population size: 100
- Random state: 42
- Training size: 75%

Exported pipeline: GradientBoostingRegressor(alpha=0.8, learning rate=0.1, loss="huber", max depth=10, max features=0.45, min samples leaf=3, min samples split=2, n estimators=100, subsample=1.0)

At this point we discover an error with our random state number generator for the pipeline search. This bug is reported for TPOT and seems to occur in OSX environments which is the case in this research. After deeper investigation the random state does work to split the data into training and test. The random state parameter also works for the final fit on the exported pipeline. The problem only occurs for the pipeline search which is not an issue as long as we use the same training and testing sets results are comparable only the pipeline search gets randomized.

Pipeline results:

- Average CV score: -651
- TPOT score function: 0.98680
- RMSE with 75% training and 25% testing data: 17.72
- RMSE with 85% training and 15% testing data: 14,63
- RMSE with 65% training and 35% testing data: 24.00

We obtain a new best RMSE for a 85% training size. We notice a second time in a row that the 65% training size results in a bad score due to underfitting.

4) Gradient Boosting Regressor 2: Due to the fact the random state does not affect the pipeline search we execute another pipeline search with identical parameters.

Pipeline search parameters:

- Generations: 100
- Population size: 100
- Random state: 42
- Training size: 75%

Exported pipeline: GradientBoostingRegressor(alpha=0.85, learning rate=0.1, loss="huber", max depth=8, max features=0.8500000000000001, min samples leaf=3, min samples split=5, n estimators=100, subsample=1.0)

The results is a similar pipeline compared to previous with different hyperparameters.

Pipeline results:

- Average CV score: -652
- TPOT score function: 0.97799
- RMSE with 75% training and 25% testing data: 18.15
- RMSE with 85% training and 15% testing data: 20.89
- RMSE with 65% training and 35% testing data: 25.10

5) Gradient Boosting Regressor 3: We increase the pipeline search by increasing the number of generations and population size.

Pipeline search parameters:

- Generations: 200
- Population size: 200
- Random state: 42
- Training size: 75%

Exported pipeline: GradientBoostingRegressor(alpha=0.8, learning rate=0.1, loss="huber", max depth=10,

max features=0.6500000000000001, min samples leaf=3, min samples split=3, n estimators=100, subsample=0.9500000000000001)

Pipeline results:

- Average CV score: -548
- TPOT score function: 0.98653
- RMSE with 75% training and 25% testing data: 16.67
- RMSE with 85% training and 15% testing data: 21.06
- RMSE with 65% training and 35% testing data: 18.78

A pipeline with similar results to the current best pipeline, XGB2. Although this is one of the best pipelines we expected more improvement from the this extensive pipeline search.

6) Gradient Boosting Regressor 4: Another large pipeline search with the same amount of pipelines evaluated but the number of generations and population size different from each other.

Pipeline search parameters:

- Generations: 300
- Population size: 100
- Random state: 42
- Training size: 75%

Exported pipeline: GradientBoostingRegressor(alpha=0.8, learning rate=0.1, loss="huber", max depth=10, max features=0.45, min samples leaf=3, min samples split=3, n estimators=100, subsample=1.0)

Pipeline results:

- Average CV score: -651
- TPOT score function: 0.98680
- RMSE with 75% training and 25% testing data: 17.72
- RMSE with 85% training and 15% testing data: 14.63
- RMSE with 65% training and 35% testing data: 24.00

The resulted pipeline resembles the previous 3 pipelines. The difference between the resulted pipelines differ in hyperparameters while feature selection and model selection is identical. Therefor we stop searching better pipelines and start optimizing specific parameters for our best pipelines. The focus of the parameter optimisation is on the RMSE and not the TPOT scoring function. Nevertheless we want the TPOT score to stay at acceptable ranges. When one score improves but the other decreases drastically it is a sign we our overfitting the parameters on our testing data for that particular scoring function.

7) Gradient Boost Regressor 3 with manual hyperparameter optimisation: After trial and error of the parameters from previous exported pipeline we were able to get better results. Exported pipeline: GradientBoostingRegressor(alpha=0.8, learning rate=0.09, loss="huber", max depth=10, max features=0.45, min samples leaf=4, min samples split=3, n estimators=100, subsample=1.0)

Pipeline results:

- TPOT score function: 0.97179
- RMSE with 75% training and 25% testing data: 15.99

The result is a slightly improved RMSE with a slightly descreased TPOT score.

8) Gradient Boost Regressor 4 with manual hyperparameter optimisation: For this pipeline we manually optimised the parameters for the 75% and 85% training sizes since they both have good results. The optimal pipeline is found for the training size of 85%. Exported pipeline: GradientBoostingRegressor(alpha=0.8, learning rate=0.09, loss="huber", max depth=9, max features=0.45, min samples leaf=3, min samples split=3, n estimators=300, subsample=1.0)

Pipeline results:

- TPOT score function: 0.9924
- RMSE with 85% training and 15% testing data: 13.37

This is the result with the lowest RMSE and the highest TPOT score. An overview of the different results is given in table 2 and figure 3 show the Gradient Boost Regressor 4 with optimal hyperparameter optimisation prediction results compared to the tire repairs from the test data.

pipeline	generations	population size	RMSE	TPOT score
XGB	100	100	16.67	0.99920
GradientBoost	100	100	17.72	0.98680
GradientBoost2	100	100	18.15	0.97799
GradientBoost3	200	200	16.67	0.98653
GradientBoost3Optimal	200	200	15.99	0.97179
GradientBoost4	300	100	14.63	0.9802
GradientBoost4Optimal	300	100	13.37	0.99240

TABLE II
OVERVIEW OF PIPELINE RESULTS

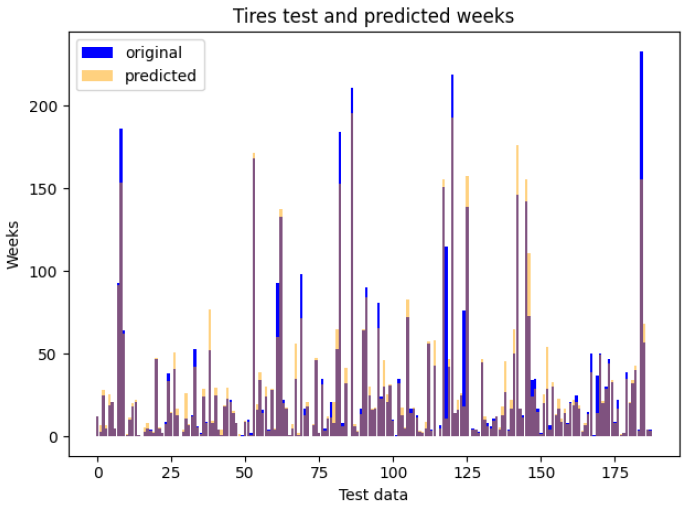

Fig. 3. Gradient Boost 4 with hyperparameter optimisation results

V. DISCUSSION

The results show that it is possible to optimise maintenance schedules with AI and ML techniques. Our best RMSE is 13.37 weeks. Compared to 29.6 weeks average between two tire repairs this is a high error. If we put this PdM model in practice this would mean that on average a tire would need repair between 16.2 weeks and 43.0 weeks. To put this PdM maintenance schedule in practice the results are not

accurate enough. There are several causes why the results are not accurate enough. A first one is the fact the target only correlates with kilometers travelled, motor time and driving time and that more specific data such as tire pressure, tire temperature or weather conditions need to be included to become a more accurate PdM model. Another possible reason is that the data between two tire repairs are not for the same tires. When a tire repair is executed two tires of the same axis get replaced. This is a inaccuracy in our data we can not remove since this information is not on the maintenance invoices. A third explanation for the inaccuracy is the that a tire replacement are not always caused by tire wear but from unexpected factors such as blown tires due to hitting a curb or nails on the road. It is likely that more accurate predictions of tire wear can be made if we predict the distance before tire maintenance instead of the amount of weeks. Nevertheless predicting the amount of kilometers does not optimise our maintenance schedule since we can not predict when the truck will reach that amount of kilometers and so no appointment at the repair shop can be planned.

VI. Conclusion

This research investigated if predictive maintenance for tires is possible and beneficial based on simple activity logs of a truck fleet. The evidence of this study show that it is technically possible to predict the amount of weeks before a tire repair is needed. Even though the model does not optimise the current reactive or preventive maintenance schedule due to low accuracy the potential of PdM in the logistics sector for a fleet of trucks is shown. The main causes for the results to be insufficiently accurate is the lack of more specific tire data. A second cause is the lack of information which tire is replaced from the maintenance invoices. Future work needs to be done to establish if PdM for tires can become viable compared to preventive and or reactive maintenance. With accurate data and integration of other data sources influencing tire wear more accurate models can be achieved. If enough data is available we can also look at deep learning methods such as neural networks instead of classic machine learning techniques.

References

[1] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, pp. 36–45, 1966.

[2] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.

[3] G. Chung, B. Gesing, G. Steinhauer, M. Heck, and K. Dierkx, "Important_Artificial Intelligence in Logistics," p. 45, 2018. [Online]. Available: https://www.dhl.com/content/dam/dhl/global/core/documents/pdf/glo-core-trend-report-artificial-intelligence.pdf

[4] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A survey of predictive maintenance: Systems, purposes and approaches," 2019.

[5] Y. Wang, S. Limmer, D. Van Nguyen, M. Olhofer, T. Bäck, and M. Emmerich, "Optimizing the maintenance schedule for a vehicle fleet: a simulation-based case study," *Engineering Optimization*, 2021.

[6] S. Byttner, T. Rögnvaldsson, and M. Svensson, "Consensus self-organized models for fault detection (COSMO)," *Engineering Applications of Artificial Intelligence*, 2011.

[7] T. Rögnvaldsson, S. Nowaczyk, S. Byttner, R. Prytz, and M. Svensson, "Self-monitoring for maintenance of vehicle fleets," *Data Mining and Knowledge Discovery*, 2018. [Online]. Available: https://doi.org/10.1007/s10618-017-0538-6

[8] P. Killeen, B. Ding, I. Kiringa, and T. Yeap, "IoT-based predictive maintenance for fleet management," in *Procedia Computer Science*, 2019.

[9] A. Holst, M. Bohlin, J. Ekman, O. Sellin, B. Lindström, and S. Larsen, "Statistical anomaly detection for train fleets," in *AI Magazine*, 2013.

[10] S. Umeda, K. Tamaki, M. Sumiya, and Y. Kamaji, "Planned Maintenance Schedule Update Method for Predictive Maintenance of Semiconductor Plasma Etcher," *IEEE Transactions on Semiconductor Manufacturing*, vol. 34, no. 3, pp. 296–300, 2021.

[11] Q. Wang, S. Bu, and Z. He, "Achieving Predictive and Proactive Maintenance for High-Speed Railway Power Equipment with LSTM-RNN," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6509–6517, 2020.

[12] Transics, "Transics api," last accessed 3 December 2021. [Online]. Available: https://integration-api-doc.euwe1.shared.cvcs.zf.com/#/

[13] TPOT, "Tpot examples," last accessed 20 December 2021. [Online]. Available: https://epistasislab.github.io/tpot/examples/