Predictive maintenance

“The alternative [to thinking ahead] would be to think backwards … and that’s just remembering.”
— Sheldon, the theoretical physicist on The Big Bang Theory

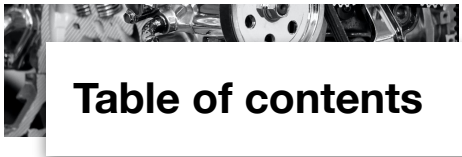


Table of contents

I. Business understanding

Every day, we are using more and more so-called smart devices. And now, in the time of the worldwide pandemic, we rely on them more than we have ever done before. While those devices enable us to work in ways that would be unbelievable just 10 years ago, they collect enormous amounts of data. To better explain this principle, let's use your average Steve for example. Every morning, Steve is woken up with a smart clock. After his breakfast, he takes the quickest route to his workplace following the instructions from the navigation application on his smartphone. Since today is his birthday, he decides to treat his colleagues with a cake. He knows that his friends deserve only the best cake for his 25th birthday so he Googles for the best cake in town. What he doesn't know is that his wife and friends prepared a surprise party for him in the evening. He was so thrilled when he found out that he immediately posted an image of the gathering on the Instagram tagging the restaurant they were at. If you have been closely following Steve's day, you might've noticed that his smartphone knows more about him than he would probably like to admit. The same can be said for a lot of us. Devices have become so good at collecting our personal information that we don't even notice it until we take a closer look. The natural question then becomes why. Why do we let these companies collect our information and what do they use it for? Since we are not going to go deep into this subject, we'll just answer with the easiest cause there is. We let them use our data because, in turn, we get to use their services like navigation and others. From the companies perspective, it helps them generate more revenue.

But people are not the only ones we can collect data from. More and more we are collecting data from various gadgets and machines, whose permission we thankfully don't have to get. And these machines are all around us in the form of IoT devices and many more. They all have a number of sensors integrated within them. The data collected can help us solve a lot of different problems. The one we are going to be exploring today is maintenance. To better understand the problem of machine maintenance we are going to take a closer look at Steve's workplace.

Since we still haven't told you what Steve does for a living, now is a good time to tell you about it. Steve is a manager at a local storage facility. As you might have guessed from the birthday example above, his employees absolutely love him and he tries to treat them as friends as much as possible. Not only is Steve a great boss, but he is also a very smart manager with unprecedented business skills not often seen in such small towns. About two years ago, his business associates told him that it might be a good idea to install a variety of sensors on the forklifts in his facility. At first, he didn't understand why, but he knew that these associates were some of the smartest people he knew so he decided to go for it. Recently, he finally understood what the fuss was all about. Something just clicked in his head and he realized what a genius idea installing the sensors was. To better explain the brilliance behind the idea, we need to talk about numbers[1]. Steve's facility owns 7 forklifts, each of which requires regular maintenance. The average service costs about let's say 1,000$ and lasts for a week. During that week, the forklift can't be used so including the opportunity costs, the total cost rises to about 10,000$ for each service, and there are 7 such vehicles. The logical conclusion is that Steve would like his forklifts to be serviced as rarely as possible. Now would be a great time to introduce you to the cost of the average automatic forklift Steve uses. It's 200,000$. By now, you have probably started getting a grasp of what the problem is. Steve would like his services to be as sparse as possible, but still not sparse enough so that the forklifts break and he needs to buy new ones. The solution for his problem are the sensors he installed on his forklifts about two years ago.

You see, the sensors measure shocks on various parts of the machine. With the enormous amounts of data he has collected throughout the past two years he can fix his problems. He decided to hire a couple of data scientists who managed to analyze his dataset and create a model that can predict when each of his 7 precious machines is about to fail and schedule them for service. It has enabled Steve to have the best of both worlds, sparse timing of maintenance and machines not breaking. Since the computer is predicting when the maintenance is needed, this is called predictive maintenance and is exactly the problem which we'll be covering today.

---

[1] The numbers were made up and used figuratively to illustrate a point.

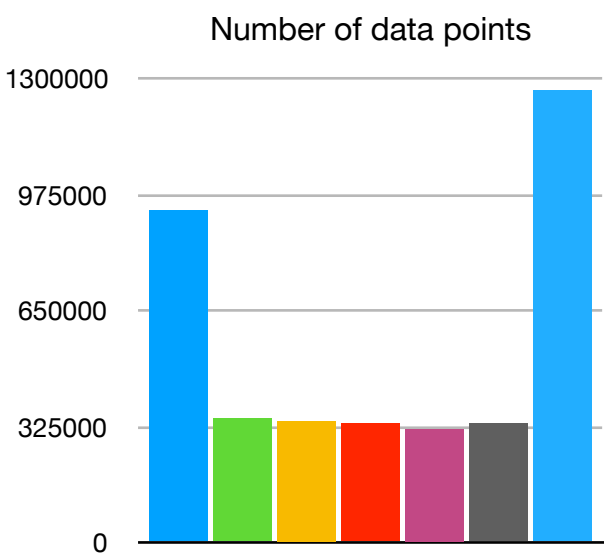
II. Data understanding

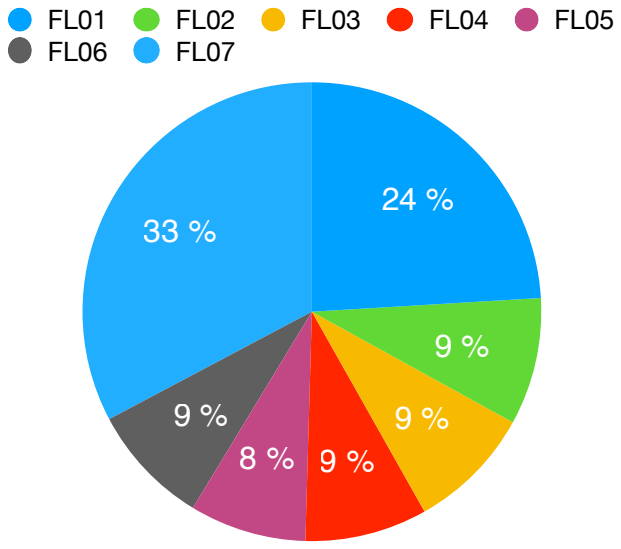
Our task is in fact quite similar to the one Steve has given his data scientists. Atomic Intelligence d.o.o. gave us the data from 7 forklifts, each of which has 6 sensors from various targeted parts of the machine. All of the data was put together in one single CSV file totaling about 400 MB. As you can imagine, it's basically a giant mess. When we first opened the file, it was like we walked into the offices of Stratton Oakmont (from The Wolf of Wall Street) right after their sudden growth spurt following the Forbes article about them. We felt overwhelmed and didn't know where to start, it felt like the first day of college. In the end, that perspective actually really helped us. The question was, what did each one of us do on their first day. The answer is obvious, we started exploring what was around us. Those are the exact principles we followed at the beginning of this analysis. We found that the given CSV file contained 7 different columns, each containing valuable information.

The dataset (CSV)

Columns	Machine name	Sensor type	Date measurement	Start timestamp	End timestamp	Realvalue	Unit

We continued with the initial inspection and found that indeed we had information from 7 different forklifts indexed from FL01 to FL07. Unfortunately, not all of the machines collected the data for 2 years like the ones from the Steves example. Here's what we found.





Machines FL01 and FL07 collected way more data than others so we decided to quantify that and see exactly when each machine started and ended collecting data.

Machine collection timeframes

Machine	FL01	FL02	FL03	FL04	FL05	FL06	FL07
Start	02.09.2017.	19.03.2019.	19.03.2019.	19.03.2019.	19.03.2019.	19.03.2019.	08.09.2017.
End	24.7.2019.	16.7.2019.	26.7.2019.	24.7.2019.	24.7.2019.	24.7.2019.	10.7.2019.
Length days	690	119	129	127	127	127	670

At this point, as you might have already noticed yourself, we came to an unexpected conclusion. FL07 has quite more data points compared to FL01 while collecting data for less time overall. Upon further inspection, we realized that there were days when machines weren't collecting any data whatsoever.

Contrary to what was said in the task description, machines weren't making three measurements per day, but the number fluctuated. In addition, we decided to take a look at how long each measurement had taken to be made. We were delighted to see that the average time for a measurement was only about 1 second because that enabled us to omit certain columns, but for now let's not get too far ahead of ourselves.

Measurement duration

	Mean	Standard deviation	Minimum	Maximum
Time	1s 251ms	3m 58s 543ms	0t	3d 3h 20m 26s 417ms

Even though there were only 6 sensors on each machine, we found that there were 12 distinct sensor types present in the dataset which complies with the official documentation given to us which states that each sensor measured both the effective velocity of vibrations (measured in mm/s) and the maximum acceleration of shocks (measured in mg[2]).

Sensor types

Lifting motor	Lifting gear	Drive motor	Drive gear	Drive wheel	Idle wheel

What followed was a task we never anticipated to be as fun as it turned out to be. We looked at graphs of each sensor data of each individual forklift. That's 12 sensor data per machine, which brings the total up to 84 graphs we had to look at. And what a journey it was. Not to bore anyone with lots of information, we are only going to state the most important observations we made while looking at them.

The first one is some kind of event around the middle of March 2019. It was crystal clear that something was wrong with the data during that time on the machines FL01 and FL07. Following the communication with the company that provided us with the data, we found out the reason behind the change. Apparently, there was a recalibration of certain sensors on these machines which caused the sudden change of readings.

The second observation was that there are pronounced spikes in readings on most of the sensors. To be perfectly honest, that was not something we were expecting. Nevertheless, we soon realized that with real (non-laboratory conditions) come these fluctuations in data.

Overall, we were happy with the data we were given. Although at first, it seemed cluttered and unorganized, after the initial analysis we realized that it was in reality very much organized and presented cleanly except for the occasional missing measurement.

2 $\approx 9.81 * 10^{-3} m/s^2$

III. Data preparation

Following the initial inspection, we were able to proceed with preparing the data for modeling. Seeing what we saw during the exploration phase, we decided on excluding some columns from further analysis.

Given dataset (CSV)

Columns	Machine name	Sensor type	Date measurement	Start timestamp	End timestamp	Realvalue	Unit
Do we care about it							

And don't worry, we didn't pick this at random. There is a precise reason for why we chose not to include each of the three columns. But before we explain that, let's start with the columns we ended up using during the task. It's pretty clear why we decided to include the first column, machine name. If we go back to the analogy of the first day of college, it's like remembering the first names of people we end up meeting. It's basic, but at the same vital information. The second column, sensor type, goes a little bit more in-depth, like the last name. Likewise, the date measurement and realvalue present important information which can be thought of as the birthday and age of our friends. On the contrary, we agreed to exclude the unit since each sensor type reported the data in the same unit. Furthermore, the end timestamp was omitted since most measurements had been made instantaneously.

And just like that, we have just reduced the number of data by over 28%, that's a pretty big jump. Especially considering that at the beginning we had a total of 27.023.038 distinct values (3.860.434 records * 7 values per record). Now, we are left with "only" 19.302.170 values. If you ask us, this is a big feat on its own.

In order to cope with the fluctuations in the data points, we decided to remove multiple daily values of sensors. That way we were left with only one data point per sensor per day. To do that, we calculated the daily mean value of each sensor. In turn, this allowed us to get rid of another column from the dataset, the start timestamp.

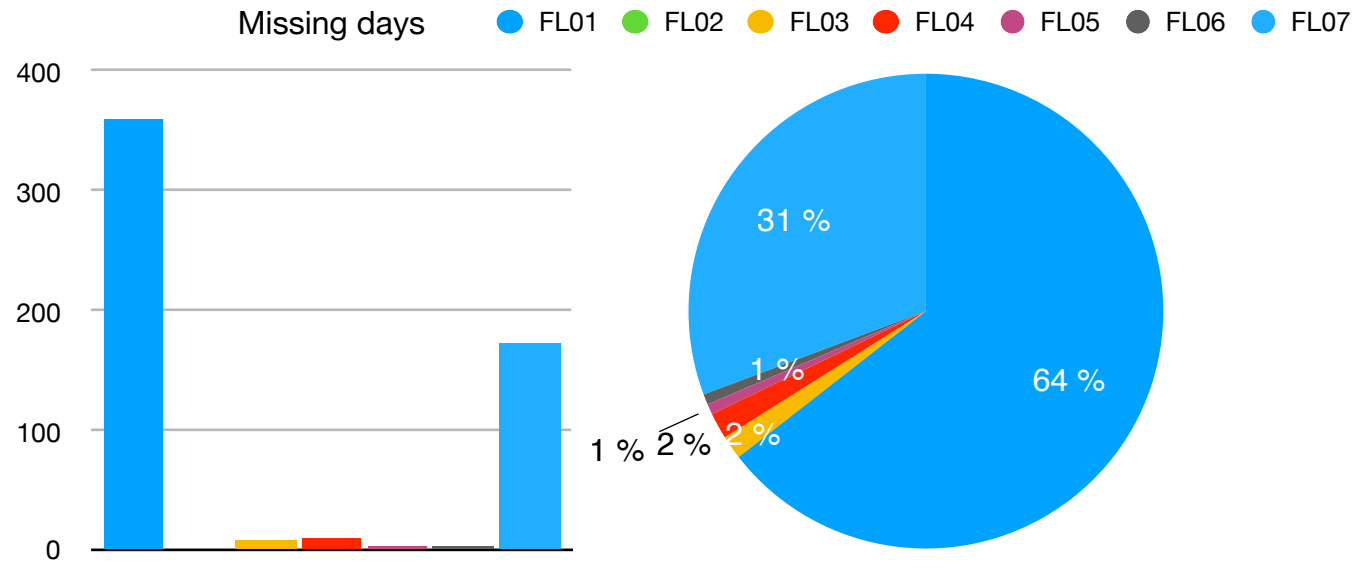
Now that we had only one value per sensor per day, we created an entirely new dataset with a new structure. First, we split the dataset into 7 different datasets forming an array. Each dataset of the array contained only the data for the designated machine. That's why there's 7 of them, for machines FL01 through FL07. Subsequently, because there was only one reading for each sensor per day, we transformed the datasets into the structure shown in the table.

New dataset structure

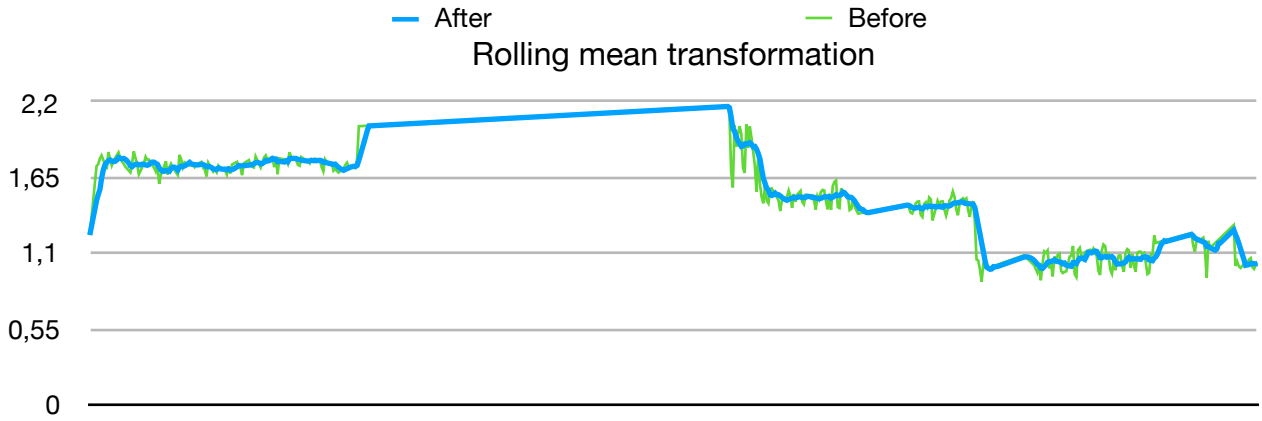
Date	**Sensor #1** Effective velocity of vibrations	**Sensor #1** Maximum acceleration of shocks	...	**Sensor #6** Effective velocity of vibrations	**Sensor #6** Maximum acceleration of shocks

The idea of having each row represent values for each sensor on a certain date has later proven to be a perfect match for a model we were building. Additionally, it provided us with a cleaner look at the available data since now we were able to easily filter data by machine, sensor, date, or any combination of the mentioned.

Closely examining the derived datasets, we realized that some of the dates were missing, which complies with the findings from the initial inspection. To address this situation, we have taken a look at several different methods for dealing with missing data like filling the values with the last known data point, filling with 0, and various kinds of interpolation. In the end, we decided to use linear interpolation which linearly approximated the missing values.



Unfortunately, after taking one more look at the graphs, it could be seen that the fluctuations were still very much present in the dataset. For that reason, we decided to apply the mathematical transformation of rolling mean for the previous 7 days on each data point. What that means is that the value of each sensor becomes the mean of the past 7 values for that sensor. As can be seen on the graph for one of the sensors, the values no longer had these giant fluctuations, but at the same time, they weren't smooth either. If the graphs were smooth, we would probably have lost valuable information hidden within them.



Now that we were mostly happy with the data we had come up with, we had to apply another transformation. At first it might seem unnecessary, but the difference it makes during the modeling phase is night and day. What we did was we applied the required transformations to get the standard normal distribution of the dataset. All the parameters of the transformation were saved so they could be later applied during the deployment stage to get the most accurate results.

To deal with the unexpected behavior of the sensors during the times of recalibration, we settled on removing these data points since they present behavior that is not normal and shouldn't be seen during normal use. This required us to make another split of the data. Therefore we were left with an array of 9 datasets, one for each machine FL02 through FL06, and two for FL01 and FL07.

Array of datasets

FL01 Before service	**FL01** After service	**FL02**	**FL03**	**FL04**	**FL05**	**FL06**	**FL07** Before service	**FL07** After service

At last, we need to address one more thing. Along with the dataset, we were provided with the dates when each of the machines went into service. During those times we aren't able to completely trust the sensory reading. But still, in an effort to potentially train our model to see these behaviors as normal, we decided not to leave out this data.

IV. Modeling

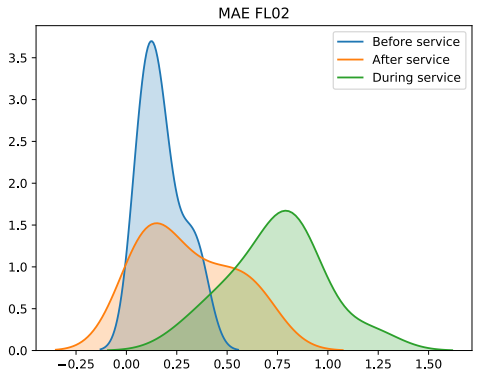
The target outcome of this project is predictive maintenance. This got us thinking for a long time about how we should tackle this problem. And it was no easy task. We looked over the data numerous times and after a while made a conclusion that a machine that is in desperate need of service should not be acting normal, yet anomalous. By acting normal, we mean the sensory readings shouldn't be normal. Thereafter, another question was raised: what is normal behavior? Previous services were made on a regular basis, not necessarily at the most optimal time. For that reason, there was no way for us to know what that different, in need of service, data looked like. The only fact we knew for sure is that most of the data present normal machine behavior. This meant that supervised learning was out of the picture, or so we thought at that moment. We started with creating unsupervised models, but at later point figured out that there was still hope for a supervised learning model since the unsupervised ones hadn't given us satisfying results. To better explain exactly what we've done, you must meet our good friends Harvey and Mike.

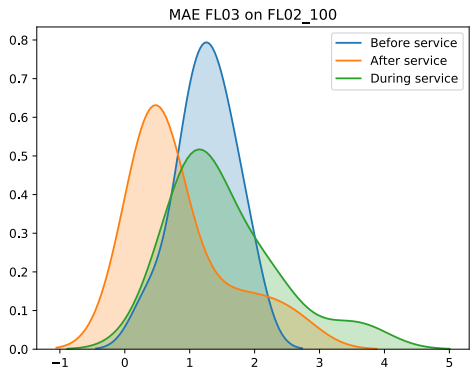
A. Mike the memory genius (Autoencoder)

Michael James "Mike", there is a lot to be said about him. He had a mind of a genius and could represent complex patterns in unbelievably simple ways. To put it simply, Mike is an autoencoder. By now, you may be asking yourselves what is an autoencoder and how can it help us achieve our goal? The autoencoder is a deep neural network which learns hidden patterns in data. It learns efficient data encodings in an unsupervised manner. In a way, it tries remembering something, and then it tries to recreate what it remembers. That was a lot of complex terminology, but we are sure everything will fall into place when we go over the example of what Mike did.

Once upon a time, Mike started working at a warehouse in Atlanta. His job was to monitor a number of forklifts used to relocate pallets throughout the facility. As we by now know, from the business understanding section, the expenses of forklifts breaking down can be huge for the company. He spent his days looking at the graphs of sensory readings from the forklifts. After a while, he started recognising particular patterns in the data that represented healthy behavior. Therefore, whenever he felt like something wasn't normal, he would file a report to his superior. Because his boss knew that Mike had understood the data inside out, the corresponding forklift would undergo service. Now that everyone has a general idea of how the autoencoder can help us, let's see exactly how we used it to detect anomalies in the data.

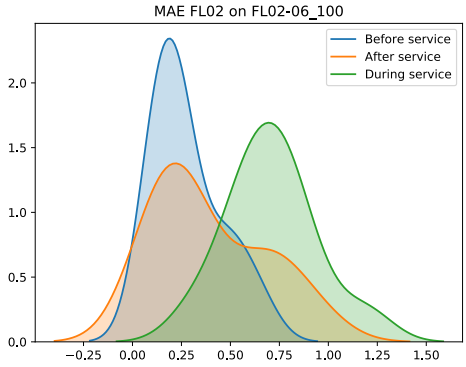
Our first idea was to create 7 separate autoencoders, each for its own machine. Since the output of the autoencoder is the same as the input because the model basically tries to recreate the input data, we wanted to measure the error of the output and use that metric to decide whether the data was anomalous. If there was a big error on a piece of data, it meant that the given data presents out of the ordinary behavior. Training the models is really simple, we just feed them the data from the past X days and that's it. After some tweaking, due to the size of the dataset, we found that we got the best results when we fed it the last 7 days of data from all sensors. When the training for all machines was complete, we got to testing and seeing how our models performed on the parts of the dataset we knew presented unusual conduct, like the times of service. At first glance, the results were extraordinary, there was a clear cut between errors on normal and abnormal data.

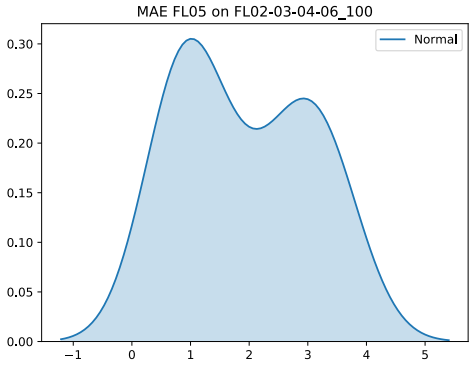




Then we decided to evaluate the model trained on the FL02 with FL03's data. What we found was that there was still somewhat of a split between the data but at the same time a lot of questions were raised. If the model can't discern abnormal data from a different machine, there must be a great chance that it won't be able to complete the task on its own machine after some time. And that's not a good thing.

To combat the problem, we decided to train the model on multiple machines and then see what happens. The outcome is unfortunately not good. At first sight, we managed to get a well-defined split on the FL02's data. Digging deeper, and evaluating other machines we could only find bad news. For example, machine FL05, which was never in service, gave us errors which the FL02 doesn't come close to, not even during service.



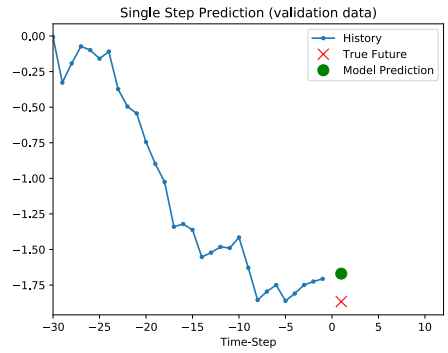


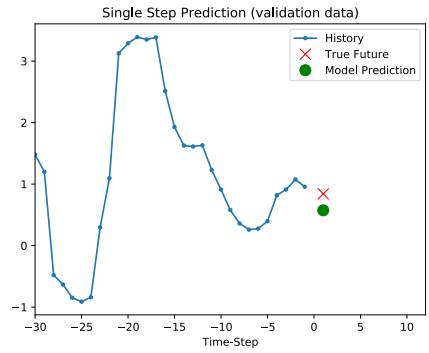
For these reasons, we decided not to proceed with pursuing the autoencoder path. Additionally, we realized that we could try and predict the future readings of the sensors and go from there. That's exactly what we did. Get ready to meet Harvey!

B. Harvey the forecaster (RNN)

Harvey Specter, our recurrent neural network, was unlike any other. We would give him data which a certain forklift reported in the past 30 days and then he could, with admirably high precision, predict what today's readings would look like. It's just unbelievable. But what's his secret? Only hard work really, he spent his days studying data from the past to become the best at his job. And he succeeded! Now that the general idea is clear, we will tell you how we accomplished this.

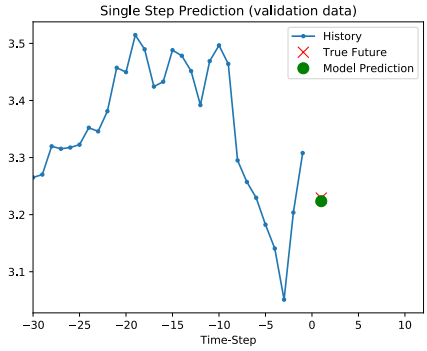
We started by creating 12 models per machine, 84 in total. It's important to note that we created these models in such a way that they are aware of the order of which the data is given to them. They understand that Monday's data is followed by Tuesday's and so on. Some people might be asking themselves why 12 models and not just one. The reason behind the number 12 is that we wanted to predict each sensor individually. Since there are 12 sensors, we have 12 models per machine. Before we started training, we split the dataset. The decision was made that the ideal split is 70/30 which means that we would use 70% of the data for training and 30% for validation. Then we started training, and what a process it was due to the sheer number of trainable parameters of the model. The model we designed has over 300,000 of them. When it finally came time to see how the model does on previously unseen data, we were taken aback. The model exceeded our expectations by far. We never thought that it can become so good at guessing what the next reading would be.

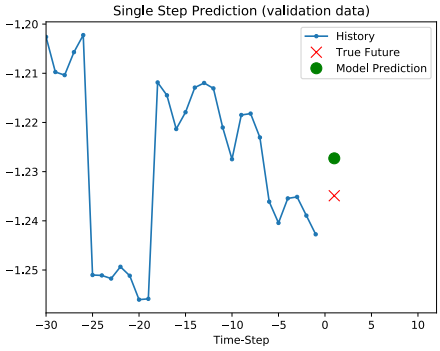




After the great results, we decided to take a step back and see how we could improve the model. What we did was very simple and in a certain way, something we learned from our past mistakes. To make sure that we get these kinds of results even far into the future, we decided to profit by using the technique called transfer learning.

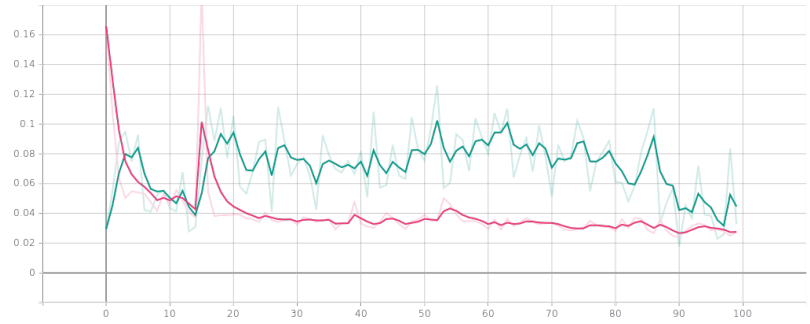
First, we created 12 models (one for each sensor) and train them on the whole dataset with all the machines present. In the end of this step, we had 12 models that are great at generalizing and predicting in a wide variety of circumstances. Unfortunately, they weren't as good at predicting a specific machine as they were generally. This is where the transfer learning practice comes into play. What we did is take these 12 models to create 84 models following a simple procedure. We took the 12 general models and replicated them. Subsequently, we further trained them on a designated machine. What we were left with is 84 models which were not only great at predicting their own data but also at predicting various patterns previously unseen on their own machines. The following are some of the results we got from these models.

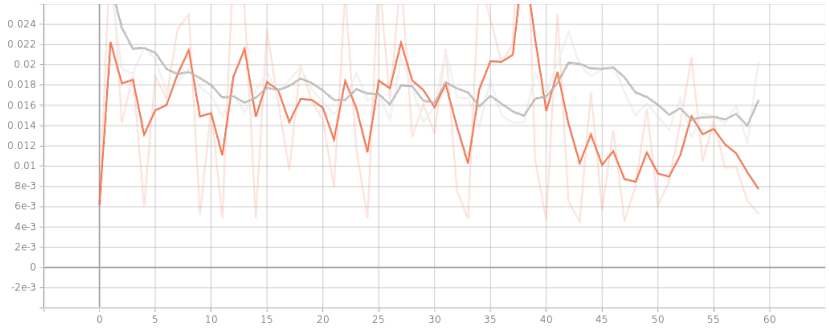




V. Evaluation

We have come to a point where we have working models that can predict the future behavior of the machines. But we are not done yet since we have 2 different options. We can either use the models which were individually trained on each machine or the models which were, at first, trained on the whole dataset, and then on the individual machines. As we said before, we believe that the transfer trained models should perform better. To verify our hypothesis, we fired up Tensorboard[3].





We found that, generally, transfer trained models performed better both on training and validation data[4]. Subsequently, knowing this and taking into account that transfer trained models should theoretically be better at generalizing, we decided to use them for deployment.

What was left for us to do at this point is to decide how we are going to use this model for predictive maintenance. Our vague idea was that we should somehow inform the staff whenever we felt like the maintenance was necessary. The question was then, how do we know when the machine is in need of service. The best answer we came up with was to monitor the prediction error each day for all of the sensors. Then, when the error exceeded the double model's mean error for three subsequent days, we would be certain that something was not right with the machine. That's the point when the machine should, under all circumstances, be serviced because of its abnormal behavior.

A. Next steps

Of course, this is not the end. Depending on the deployment structure (where it is going to be deployed, on what kind of machine), we may still have some work to do. Because we used the Tensorflow library for modeling, we can deploy our models on a wide variety of devices ranging from small embedded devices to servers. Should our models be deployed on these small embedded devices, we still need to convert them to Tensorflow lite models which are highly optimized to work on such devices with a small amount of power.

---

[3] TensorBoard is TensorFlow's visualization toolkit, enabling you to track metrics like loss and accuracy, visualize the model graph, view histograms of weights, biases, or other tensors as they change over time, and much more. More information (graphs and links to Tensorboard) can be found in the appendix.

Furthermore, to keep our models running smoothly and correctly, we need to retrain them once in a while. This will keep them up to date with any new patterns with normal behavior monitored on said machines.

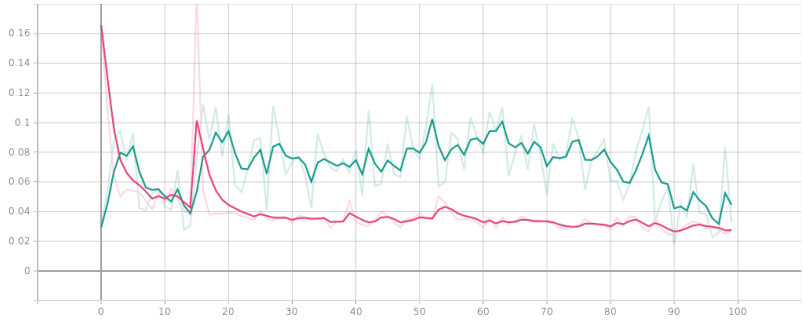
Appendixes

Appendix A

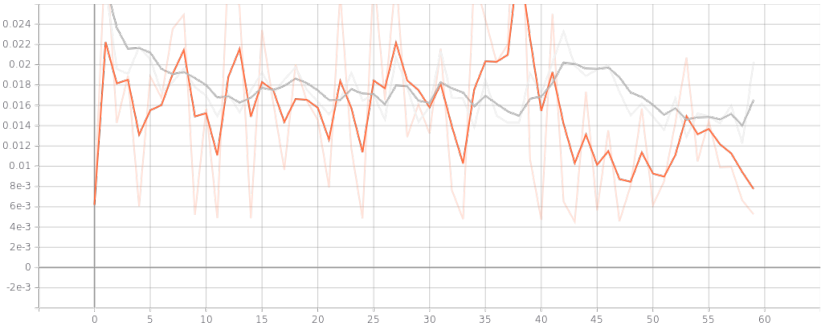
Tensorboard graphs and links to designated dashboards.

Links

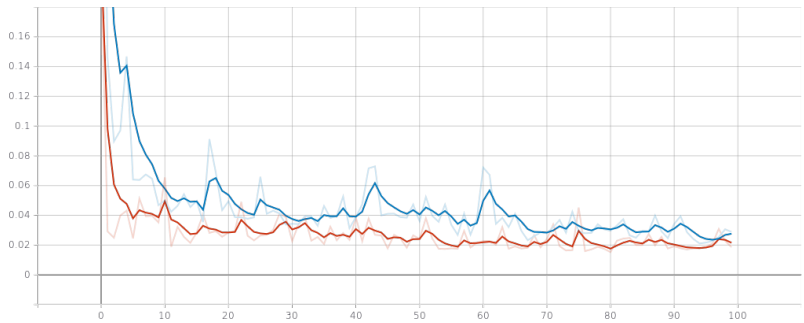
- Single machine RNN models: https://tensorboard.dev/experiment/2A06r70WQQKTxqLIM-TY6Qw/
- General machine RNN models: https://tensorboard.dev/experiment/1u1zPOcWQGGV2d-XIPXo6xw/
- Transfer single machine RNN models: https://tensorboard.dev/experiment/TNUzVVI4Rpep-VRxfYsGGPQ/



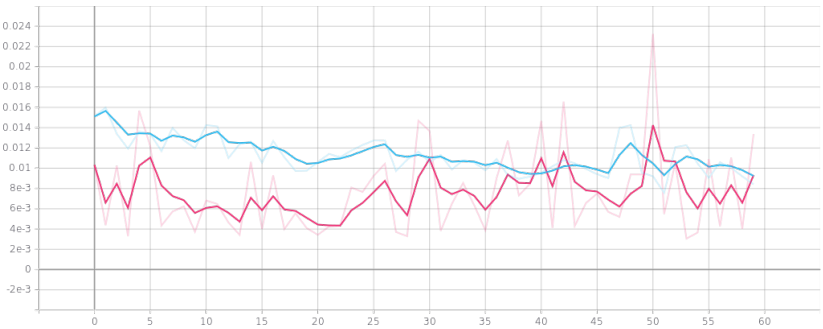
MODEL
Single machine trained
MACHINE
FL02
SENSOR
drive_gear_V_eff
METRIC
Root mean squared error
DATA
Training and validation



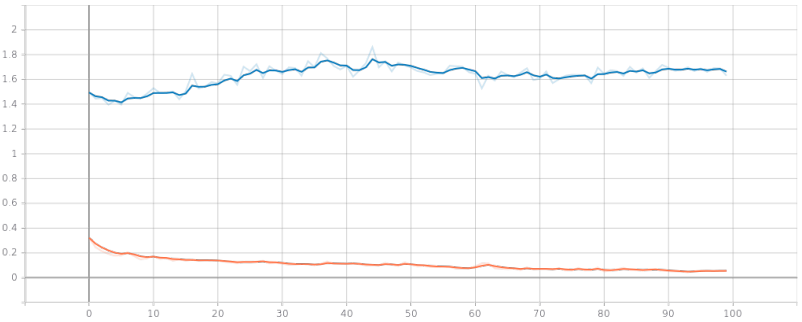
MODEL
Transfer trained
MACHINE
FL02
SENSOR
drive_gear_V_eff
METRIC
Root mean squared error
DATA
Training and validation



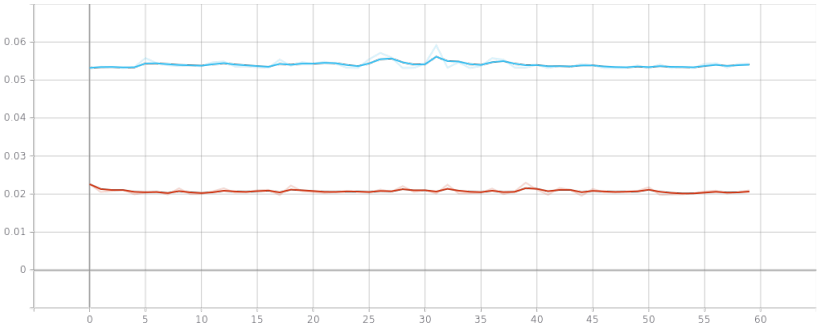
MODEL
Single machine trained
MACHINE
FL03
SENSOR
drive_wheel_V_eff
METRIC
Root mean squared error
DATA
Training and validation



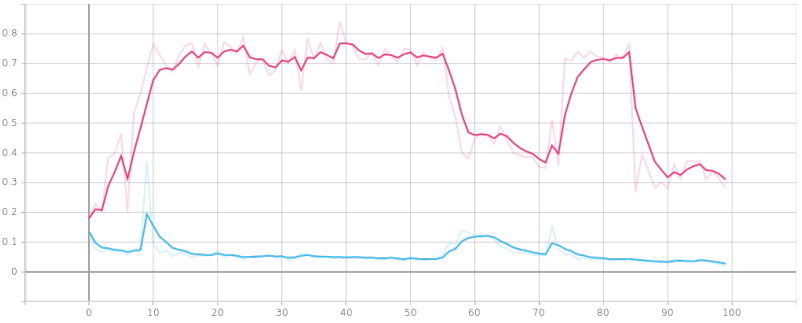
MODEL
Transfer trained
MACHINE
FL03
SENSOR
drive_wheel_V_eff
METRIC
Root mean squared error
DATA
Training and validation



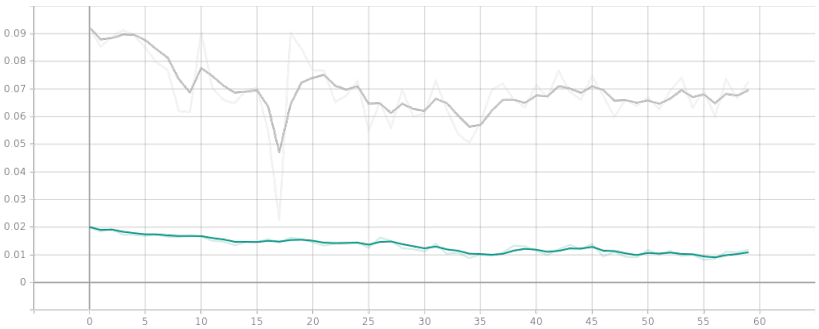
MODEL
Single machine trained
MACHINE
FL04
SENSOR
drive_motor_V_eff
METRIC
Root mean squared error
DATA
Training and validation



MODEL
Transfer trained
MACHINE
FL04
SENSOR
drive_motor_V_eff
METRIC
Root mean squared error
DATA
Training and validation



MODEL
Single machine trained
MACHINE
FL05
SENSOR
lifting_gear_a_max
METRIC
Root mean squared error
DATA
Training and validation



MODEL
Transfer trained
MACHINE
FL05
SENSOR
lifting_gear_a_max
METRIC
Root mean squared error
DATA
Training and validation