Machine Learning Predictive Maintenance on Data in the Wild

Adrian Binding
Cognitive Solutions and Innovation
Zürich, Switzerland
adrian@cogsi.ch

Nicholas Dykeman
ETH Zürich
Zürich, Switzerland
dykemann@ethz.ch

Severin Pang
Cognitive Solutions and Innovation
Zürich, Switzerland
severin@cogsi.ch

***Abstract*—In this paper, we report on our real-word experiences in forecasting machine downtime based on real-time predictions of imminent failures. Predictions are based on the use of a machine learning classification algorithm trained on historical machine data. This is constrained by the available sensor equipment. We report on our recent collaborative work with a machine builder of premium printing equipment for purposes of predictive maintenance. We describe our data analytics approach with a view towards processing unstructured data, show initial results, discuss issues and lessons learnt.**

***Index Terms*—Predictive maintenance, Internet of Things, Big data applications, Printing machinery, Statistical analysis**

I. Introduction

The Industrial Internet of Things (IIoT) is expected to deliver the next evolution in manufacturing efficiency by allowing real-time optimisation of machine assets and processes. Many of the anticipated efficiency gains are assumed to be based on a reduction of unplanned machine downtime. Continuous real-time data should give early warning of such impending faults, allowing regular maintenance actions to be scheduled, thus reducing or eliminating unplanned downtime events.

With the ubiquity of network connections and the increasing acceptance of data acquisition as well as the decreasing cost of data storage, the collection of such data is increasing. What is less obvious with currently deployed data acquisition sensors and infrastructure – which do not yet fully exploit recent advances in processing, communications, and storage capabilities – is that the available data stock is sufficient to produce status predictions which allow sufficiently accurate forecasting of fault events to materialise the anticipated gains.

In cooperation with an industrial partner, we have obtained operational data for a large central imprint printing press. Using this data, we developed an algorithm to predict certain failure events that lead to machine downtime, using only the data collected by currently installed equipment. Here, we will expand on our approach, the challenges encountered, and the lessons learned.

II. Background

The machine we studied has a small number of print-units; each of which can add a single color to the print. Photopolymer printing plates contain a mirror relief image of the required print. Ink is applied to the raised parts of the plate from which it is transferred onto the substrate, which can be wood-pulp based, synthetic or laminated material.

Ink is pumped through the system, applied to the anilox roller, and transferred onto the printing plate via contact. Excessive ink is raked off the anilox roller with a doctor blade - a long knife-like blade of steel or similar material which scrapes off excessive ink.

The ink flow system is controlled by PID controllers to guarantee proper ink flow speeds and pressures. Due to pressures, high temperatures, and the flammability of the ink fumes, the ink flow environment has to obey special safety constraints. Retroactively adding sensor equipment to the machine would incur significant costs, if it is even possible in the first place.

The printing machine is used to perform a large variety of small-batch printing jobs. This means a high rate of planned down-times to change jobs, inks, or printing plates. In addition to these operational down-times, there also are regular periods of machine maintenance. Nonetheless, equipment (and operator) failure is a frequent occurrence, leading to unplanned downtime of significant economic impact, ranging from loss of revenue, penalties, to loss of future business.

The printer is equipped with various sensors to monitor pressures, temperatures, tensions, ink flows, as well as ambient environment parameters. These variables are collected minute-by-minute and transferred into a centralized data-base infrastructure for operational purposes, e.g. OEE calculation. Additionally, equipment such as the PID controller mentioned above collect further variables significant to their operations.

Due to the difficult operating environment, data collection is not flawless. We observe missing data due to failures in the data collection and/or transmission processes, which, given the physical distribution and operational complexities of such processes, are to be expected and potentially hard and costly to remedy.

III. Data

Data was obtained from a centralized, cloud-based data repository [1]. Two information types can be retrieved as time series:

- *down-time data*: For each down-time of the machine, an entry records

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

  - the starting and end time,
  - major and minor error codes and
  - an operator entered free form annotation which briefly describes the reason for the down-time and steps taken to remedy any issues with the machine.
- *process data*: every minute a set of about 100 operational variables regarding the ambient environment (temperature, humidity, heat-index), temperatures inside the machine, tensions on the print substrate, ink flow speeds and pressures, line-speed, etc. are collected and recorded together with the sampling time as one entry in the process data table.

This data is collected historically for the purpose of understanding the machine's operational status, and not for the purpose of predictive maintenance. It was therefore not clear a-priori what variables (if any) can be used to effectively predict machine failures.

The data exhibits missing values and some obvious measurement errors. One can partially recognize these as missing values, extreme outliers, or negative measurements of non-negative variables. We dealt with these values by linear interpolation.

Furthermore, operator actions had a clear impact on the data. As an example, at semi-regular intervals, the process was slowed down, so that the operators could splice a new web to the current web. This leads to random shifts in the data based on the operator actions as opposed to the machine status as another form of potentially unpredictable noise.

IV. Prediction Algorithms

Our goal was ultimately to predict machine failures. This problem is a specific instance of *survival analysis* [2]. The goal is to either predict the remaining time until a certain event, i.e. in our specific case the *time to failure*, or whether or not an event will occur in a given time-window. The first case is an instance of *regression*, the latter of *classification*.

We focused on the classification problem to notify the operator to react when a machine failure becomes likely. To give the operator time to react, we have to ensure a sufficient *prediction horizon*, i.e. how far into the future we predict an occurrence.

A. Mathematical Formulation

The problem consists of random variables $y^t \in \{\text{True}, \text{False}\}$, each representing an event occurring within the time-window $[t, t+H]$, where H is the prediction horizon.

We further have a set of *feature variables* X^t, which is a random variable corresponding to the process variables up to the point t or some features thereof.

Our goal is, in a very broad sense, to describe the conditional probability $p(y^t \mid X^t)$ by some function $f : u \mapsto p \approx p(y^t = \text{True} \mid X^t = u)$, i.e. given an observation of the feature variables, what is the probability of failure. This function is found by optimizing some loss criterion such as e.g. the empirical cross-entropy of the estimated probabilities to the true distribution.

The use of this formulation depends heavily on the assumption that this distribution is invariant across observations. That is,

$$p(y^t = \text{True} \mid X^t = u) \approx p(y^s = \text{True} \mid X^s = u)$$

for $s \neq t$. Otherwise, we optimize for the average case, which, depending on the variability in the probabilities, can be far from the reality in each case.

Having a too large prediction horizon H reduces the use of a prediction as it makes the course of action unclear, whereas a short horizon will not allow the operator sufficient time to react. We decided on a horizon of 30 minutes as a reasonable time-window, as it allows the operator ample time to react, while also indicating a need for immediate action.

Our choice of which event types to predict was based on the abilitiy to identify a sufficiently distinct, clearly identifiable, and large set of these events in the down-time data using error codes and specific keywords in the operator annotations. We thus refined our goal to implement an algorithm to predict *print-unit failures*, that is, failures which could be clearly related to the failure of a specific printing unit and which are not due to operator activity or some external causes not reflected in the collected process data.

To this end, we focused on process variables associated with the ink flow to the print units (see figure 1) and features derived from these as our features X^t.

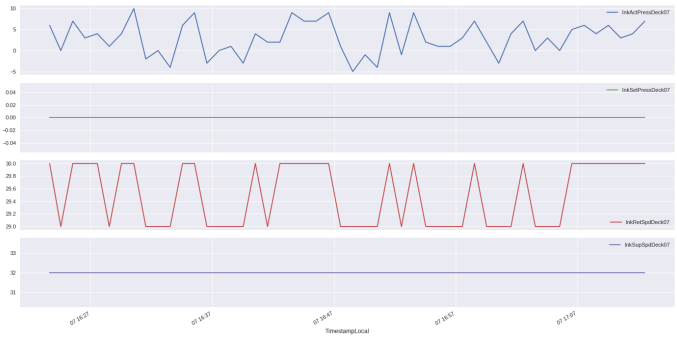


Fig. 1: Example of Ink-related variables

B. Evaluation

Evaluating classification algorithms allows for a great variety of methodologies. Standard practice uses a held-out set of "unobserved" data to compare the performance of algorithms w.r.t. various metrics and statistical tests. We withheld the data from the year 2018 in training our algorithms for this purpose. Additionally, we performed cross-validation [3] on the training data to adjust regularization parameters for those algorithms prone to overfitting to data.

This lead to the performance of our algorithms not greatly deviating between training and testing data.

We analyzed various metrics to determine the goodness of fit of our models:

- The empirical cross-entropy [4]
- The area under the receiver operating characteristic curve (AUC) [3]

- The receiver operating characteristic curve itself (ROC) [3]
- The precision-recall curve (PRC)
- The number of false positives (FP), true positives (TP), false negatives (FN) and true negatives (TN) at various decision thresholds.
- Calibration curves of the estimated probabilities.

These metrics account for the match of the model distribution and the empirical distribution in the data (cross-entropy, calibration curves), as well as their usefulness as prediction scheme, by comparing the number of false positives, true positives, false negatives and true negatives at various thresholds of predicted probability.

V. Implementation

A. Identifying print-unit failure events

Print-unit failures are a significant cause of unplanned down-time, in particular so-called blowouts, can be well identified, occur relatively frequently, and have a high operational impact.

Despite best mechanical engineering efforts, print-units produce leakages, mainly when doctor blades are worn out or excessive pressure levels build up. The result of such blow-outs are lengthy cleaning phases during which the entire printing press or at least one printing unit are not operational.

The first task was to identify which down-times corresponded to these ink unit failures. To this end, we first filtered out all relevant error codes. This subselection was then again filtered using the natural language operator annotations. Print unit failures usually contained the words "print unit", "unit", "pu" or the like.

Based on this, we started building regular expressions to identify print unit failures, and which print units were concerned. This took the form of an iterative procedure: We designed a regular expression, and trained our predictive algorithms based on the events identified by this expression. In the next step, we looked at instances where our algorithm predicted a high probability of failure which were not labeled by our current regular expression, as well as instances for which we predicted a low probability of failure, but which we had labeled as examples of a print unit failure.

This allowed us to iteratively refine our regular-expression based filter in a semi-supervised manner, continuously increasing the precision and recall. Eventually we were able to account for over 158 hours of downtime related to print-unit failure in 2017. See table I for a selection of identified downtimes. This iterative approach requires less manpower than hand-labeling all potential entries, and is similar in essence to iterative semi-supervised labeling schemes.

Loss Type	Code	Notes
Unplanned	Doctor blade	unit 10 slinging DS B&S
Unplanned	Plates-MISC	wash Lt Green plate, changed B&S ...
Unplanned	Delivery Jam	lower stacker, scrap in stacks, called ...
Unplanned	Ink System	pu 5 auto clean
Unplanned	Anilox	anilox unit 5 dried in

TABLE I: Example of filtered print-unit related downtimes.

B. Machine learning

For each point in time and each print unit, we have one of the following labels:

- The machine goes down within 30 minutes due to a failure of the given print-unit.
- The print-unit is active, and the machine remains runs for the next 30 minutes.
- The machine is down within 30 minutes, but not because a print-unit failure.
- The print-unit is not active.

This gives us up to 14 different labeled observations y^t per time stamp, i.e. one label per print-unit.

We then use those active print-unit observations where the machine has been running for at least a fixed amount of time as our set of observations to compute rolling values for means, standard deviations etc.

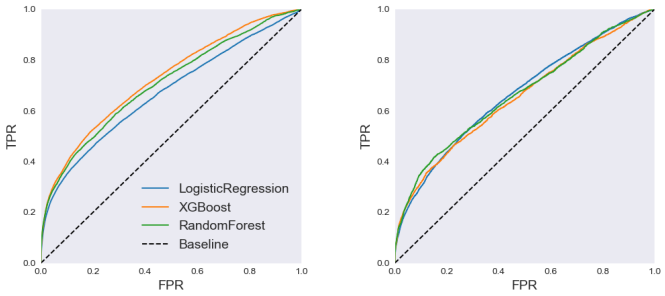
Using the observations from 2016 and 2017 to train our algorithms and the 2018 data to evaluate them, we implemented and compared several algorithms to predict the labels for each observation based on features gathered from the process data.

1) Feature engineering: For our predictive algorithm, we augmented the raw data collected with the following derived features:

- *rolling z-score*: The number of rolling standard deviations between the current value and the rolling mean. This is high for (locally) abnormal values.
- *relative change in rolling standard deviation*: The relative change in the rolling standard deviation of the data versus the previous observation. This indicates that the variation in the data is increasing or decreasing.
- *rolling autocorrelation*: The autocorrelation of each variable with itself indicates how much subsequent values correlate. If it usually does not correlate and suddenly does, this can indicate an exogenous disturbance.
- *rolling trends*: A rolling linear regression determines the linear change of the variables over time. This shows potential trends in the data, which may be caused by a disturbance.
- *power spectra*: A rolling Fast Fourier Transform (FFT) [5] of the variables gives us the energy of the variable at various frequencies. This reflects the shape of the variables curve, and is closely related to the autocorrelation. If we suddenly start seeing variation in the active frequencies, this could indicate a relevant disturbance.

This gave us a vector of 32 features per observation. Using these as our predictive variable X^t, we used state-of-the-art parametric and non-parametric classification algorithms to correlate them with the data labels.

2) Algorithm selection: Using the features defined above, we evaluated several distinct classification algorithms with a variety of hyper-parameter choices. We proceeded by selecting the best performing hyper-parameters for each model by 3-fold cross-validation on the training data in terms of the AUC



(a) train data (b) test data

Fig. 2: ROC Curves on training and test set. We see that all algorithms perform similarly.

score. The model was then fitted to the entire training set with these hyper-parameters, and subsequently evaluated on both the training and test data.

We tested the following algorithms:

1) *Logistic Regression* [3]. This does not require any parameter tuning.
2) *Random Forests* [6]. We tuned the number of trees, the maximum depth of each tree, the splitting criterion, and the number of features considered per split
3) *Extreme Gradient Boosted Trees* [7]. For these we tuned the number of trees, the maximum depth of each treet, the learning rate (relative weighting of trees added later in the process), and the sub-sampling rate of data per tree

We chose these algorithms based on their efficiency and general effectiveness in classification tasks. For Logistic Regression and Random Forests, we used the implementations in the open source Python library `sklearn` [8], for extreme gradient boosted trees, we used the open source `C++` implementation described in [9].

3) Results: From the ROC (see figure 2), we see that all algorithms perform significantly better than a random classifier which would achieve an AuC of 0.5.

From the various number of true positives, false positives and false negatives on the training set (table II) as well as the F1 score across thresholds (figure 3), we see that the gradient boosted trees and random forest clearly outperform logistic regression. For this reason, we selected the random forest in our actual implementation.

C. Online Evaluation

To analyze the effect of our prediction algorithm in practice, we compared the predictions made by our classification algorithm versus recorded down-time events in an online setting. We started recording the live data feed from machine operation, and simulated a naive operator, who would take action when the predicted likelihood of failure goes above a certain threshold.

Instances in which the operator catches a failure are *true-positives*, instances where they miss a failure are *false-negatives*, *false-positives* are those cases where a failure is

	True Positives	False Positives	False Negatives
	Decision threshold 0.05		
Logistic Regression	78	436	168
XGBoost	89	452	157
Random Forest	90	463	156
	Decision threshold 0.2		
Logistic Regression	40	88	206
XGBoost	73	147	173
Random Forest	68	100	178
	Decision threshold 0.5		
Logistic Regression	19	17	227
XGBoost	19	3	227
Random Forest	2	1	244

TABLE II: Number of classified events at varying thresholds on the test set. Positives are those events where the predicted probability of failure is above the given threshold.

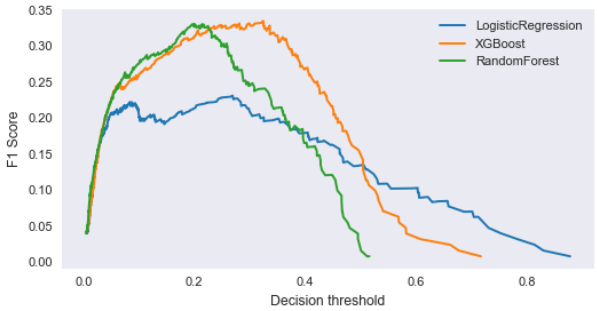


Fig. 3: F1 Score of the various classifiers over varying thresholds. We see that the Random Forest and XGBoost perform similarly well, outperforming logistic regression.

checked for but would not have occurred, and *true-negatives*, finally are then the most common case: The operator does not check for a failure and it does not occur.

We assumed that the operator checked the machine each time the algorithm predicted a likelihood of failure above 30 %. An actual failure is caught if it occurs within 10 minutes from this point in time.

True-negatives are of no interest as they do not trigger any operational action. On the contrary, *false-positives* may have an impact as operating measures, negatively impacting equpipment utilization, may be taken erroneously. They also may erode operator confidence into the predictions leading to rejection of any predictive maintenance solution. Any future operational procedures must weigh the benefits of catching a failure early on versus the cost of looking for a failure cause. The benefits of a high rate of true-positives must we weighted against the rising rate of false-positives since a near-zero rate of false-positives is preferable to a higher rate of true-positives.

In figure 4, we have depicted how the optimal decision threshold varies depending on the ratio of the benefit of catching an failure vs the cost of checking a failure. That is, we depict the optimum decision threshold under the expected benefit given a benefit to cost ratio between 0 and 100. As one can see, our threshold of 30 % coincides with a low assumed benefit-to-loss ratio. We choose this threshold based on the

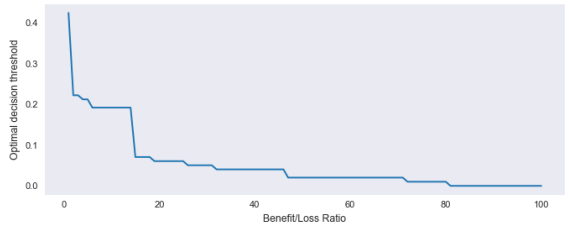


Fig. 4: Optimal decision thresholds depending on benefit-cost ratio.

additional observation that the higher precision of the resulting implementation will improve operator trust and thereby the rate of adaptation.

1) True Positives: We first show the detection of two *true-positives* on September 21, 2018.

Date & Time	Error Code	Operator Annotation
2018-09-21 11:41	Operator Error	Autoclean PU 8. Return Port clogged, stacked off 3 pallets of [PRODUCT NAME]
2018-09-21 12:46	Operator Error	Autoclean PU 2. Added Defoamer to Silver

TABLE III: True-positives on Sept. 21, 2018

The first event occurs with print-unit 8, the second event relates to print-unit 2 and indicate some issues during auto-cleansing for the given print-units.

The values of the ink flow variables on September 21, 2018 for these two print units are shown in figure 5. The vertical, dotted, lines indicate the times of the recorded failures. Prior to these points in time, the ink return speed does show some increased disturbance.

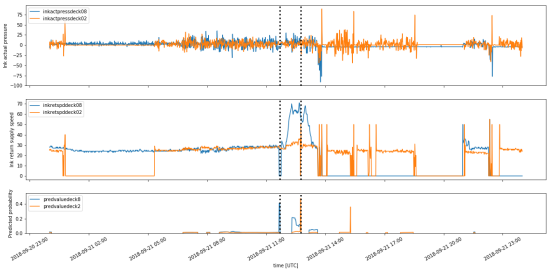


Fig. 5: Ink flow variables and prediction values for print-units 2 and 8 on Sept 21, 2018

Based on the ink flow variables and the derived features, we also show the computed prediction values at the bottom of figure 5. The spikes for the two positive predictions can clearly be identified. Note that the third spike for print-unit prediction values is not matched by a down-time event and thus would correspond to a *false-positive* - depending on the sensitivity threshold used to trigger alerts.

2) False Negatives: On September 30, 2018 we recorded a case of *false-negative* at 12:31. Operator annotations are available for that case of a false-negative and are given in table IV.

The occurrence of the non-predicted failure is indicated by the dotted vertical line in figure 6 which shows the evolution of the ink variables and the prediction values over the entire day.

Date & Time	Error Code	Operator Annotation
2018-09-30 12:31	Operator Error	Pan in PU10 flooded, changed B/S 50k. Washed plate in PU8. Cleaned off O.S. of CI; buildup causing damage to plates

TABLE IV: False-negatives on Sept. 30, 2018

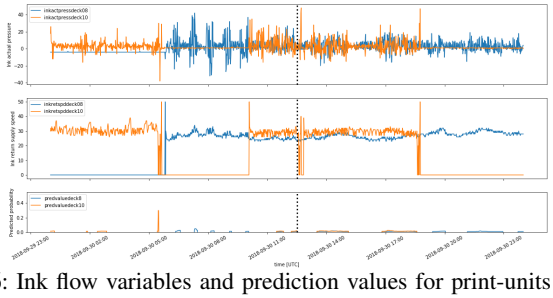


Fig. 6: Ink flow variables and prediction values for print-units 8 and 10 on Sept 30, 2018

D. Summary statistics

The total number of false-positives is 53, while we observe 28 true-positives. This corresponds to the decision threshold we set. We missed a further 24 false-negatives. Lowering our decision threshold would have caught some of these events, preventing the resulting downtimes, at the cost of causing further false-positives, with their associated costs.

VI. Discussion

As seen in the previous sections, we were able to develop a practical real time predictive algorithm using existing data sources. We here discuss some key learnings from our application and their practical implications.

A. The benefit of unstructured data

A disturbance factor in the data in general, and a hindrance in the development of a predictive algorithm, are **external, i.e. unobserved, factors**. This includes, e.g., the operator impact. By basing our predictive algorithms on only machine-related data, and ignoring the actions of the operator (up to its impact on the machine data), we are clearly missing a part of the picture.

This would be a significant challenge for any purely machine-based algorithm. Changing behaviour between shifts, intuitive actions and human error cannot be accounted for through machine data alone. Observing the operator directly and studying their impact on operations, i.e. taking operator behaviour as an additional feature, appears a promising path going forward. This can simply take the form of recording operator actions when possible. Given advances in modern interfaces, it is also becoming increasingly feasible to **gather more information from the operator to further understand their actions**.

This type of data collection, in the form of operator notes, proved invaluable when identifying failure events in our case. Even though processing free form data is non-trivial, having it at our disposal allowed us to retroactively identify specific types of events, so that we could tailor our predictive algorithm

to these. This affords a great flexibility in terms of potential applications.

B. Measure what is measurable

Another example of a **missing data point** was the PID control signal. While we had access to the controlled signals, knowing the energy going into maintaining the system stability would have been of great interest. The reason as to why this data is not available is not known to us.

With modern machinery being outfitted with ever increasing amounts of sensor equipment, we believe that this is something to keep in mind: While sensors are usually installed for a specific purpose, there is a great potential for future applications that might not be clear at the point of installation. The potential upside of data collection might not be immediately apparent, but should be taken into account early in the design process. In particular, **any data that is used in operating machinery should be readily accessible**.

C. Build systems for humans

A further key point is the **interpretation of probabilistic models in practice**. A predictive algorithm is here only as good as its impact. If it generates alarms at a probability level of 10 %, operators may believe this to be no ground for action. If one were to implement a guideline to react to these alarms regardless, operators may lose trust in the system, and ignore it entirely.

One way to prevent this would be to increase the statistical literacy of operators. Given the evidence from medical literature [10], this may prove a lofty goal indeed.

The more practical approach is to accommodate human biases in practice. This can be achieved by e.g. **favoring precision over recall**. This increases the number of identified events versus the status quo, while maintaining operator trust, allowing for a better adaptation and ensuring the success of the application in practice.

D. The future is out there

The performance of our predictive algorithm is clearly hindered by our decision not to retrofit sensors. However, such an approach would have involved retrofitting sensors in hazardous environments, which would have proved costly if not infeasible in practice. As computing power and algorithms keep developing, the possibilities to find new potential applications of data will continue to increase. Building these based on existing datasets is a relatively risk-free step towards digital transformation. For this reason, **the potential upside of retrofitted digital solutions should be a key aspect of a IIoT based design process**.

VII. Conclusion

Data mining with the hope of robust and meaningful results for prediction or failure analysis is dependent on data quality and relevance. Deploying ubiquitous, abundant and ever more performant IT resources into industrial installations with lifecycles of decades remains a challenge to unleashing the Industrial Internet of Things.

Nonetheless, existing data stocks can already provide a fruitful environment for the development of data based solutions to industry challenges, and are a sensible starting point for the development

Analysing machine data however only covers part of the entire picture: Without understanding the impact of operators on operations and understanding what motivates operator behaviour, we are missing out on a greater potential to improve operator-driven processes.

Going forward, the cost-benefit analysis of data gathering should account for the potential upside stemming from the potential for retro-fitted applications, and IIoT based applications must keep in mind the human aspect of operations if they wish to prove successful in practice.

References

[1] EI3 Corporation, "EI3 Products," https://www.ei3.com/category/products/, 2018, [Online; accessed 1-October-2018].

[2] D. W. Hosmer Jr., S. Lemeshow, and S. May, *Applied Survival Analysis: Regression Modeling of Time-to-Event Data*, 2nd ed. Wiley, 2008.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[5] W. M. Gentleman and G. Sande, "Fast fourier transforms: For fun and profit," in *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, ser. AFIPS '66 (Fall). New York, NY, USA: ACM, 1966, pp. 563–578. [Online]. Available: http://doi.acm.org/10.1145/1464291.1464352

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] Tianqi Chen et al., "XGBoost Documentation," https://xgboost.readthedocs.io/en/latest/, 2018, [Online; accessed 1-October-2018].

[10] D. Berwick, H. Fineberg, and M. Weinstein, "When doctors meet numbers," *The American Journal of Medicine*, vol. 71, no. 6, pp. 991–998, 12 1981.