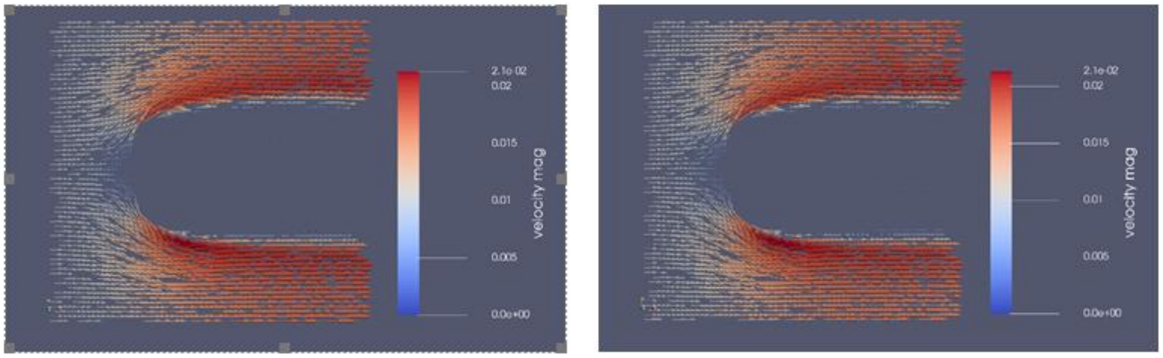
Simr Compendium of Case Studies on Artificial Intelligence in Manufacturing



Simulated flow field (left) and the 1000x faster predicted flow field (right) generated by AI.

Sponsored by:

HPC wire

May 2024

SIMR

https://www.Simr.com

Simr AI Case Studies

More than 230 HPC/CAE cloud experiments, 120+ case studies, and a ton of hands-on experience gained in High Performance Computing (HPC) in the Cloud, that's the harvest of 12 years of Simr (formerly known as UberCloud) HPC Experiments, enabling us to measure cloud computing progress, objectively. Looking back these ten years, at our first 50 cloud experiments in 2012, 26 of them failed or didn't finish, and the average duration of the successful ones was about three months. Six years later, in 2019, looking at our last 50 cloud experiments, none of them failed anymore; and the average duration of these experiments is now just about one to two weeks. That includes defining the application use case, preparing and accessing the engineering application software in the cloud, uploading the engineer's data, running simulation jobs (interactive and batch), evaluating the data via remote visualization, transferring the results back on premise, and writing and publishing a case study.

The goal of the UberCloud Experiment is to perform engineering simulation experiments in the HPC cloud with real engineering applications, in order to understand the roadblocks to success and how to overcome them, and to help **educate our engineering community** about the relatively young field of HPC in the Cloud. Our UberCloud Compendiums of Case Studies (20 e-books so far) are a way of sharing these results with our broader community of engineers and scientists and their service providers.

Our community of engineers and scientists is currently joining the new wave of artificial intelligence (AI). According to IDC the expected world-wide investment in Machine Learning is forecasted to be $97 billion in 2025, with an average annual growth rate of 44%. In addition, the three largest use cases for AI in discrete manufacturing are expected to be predictive maintenance, quality management, and recommendations systems.

This 20th UberCloud/Simr Compendium is presenting several AI cases studies of engineering simulations which we have performed between 2018 and 2023, for our own and our customers' learning purpose, but also providing several production projects use cases. For helping our engineering community to approach Machine Learning in Engineering, we have selected use cases about predictive maintenance, deep learning, computational fluid dynamics, natural language processing, and a living heart project.

We are very grateful for all the support for our UberCloud/Simr AI Experiments by our technology partners **Amazon AWS, Microsoft Azure, Google Cloud, Ansys, Dassault Systemes, Siemens, and SUSE**, and our Media Sponsor **HPCwire**.

Wolfgang Gentzsch and Joseph Pareti
Los Altos, CA, Jan 2022, updated May 2024

Please contact Simr at contact@simr.com before distributing this material in part or in full.
© Copyright 2024 Simr™. Simr is a trademark of TheUberCloud, Inc.

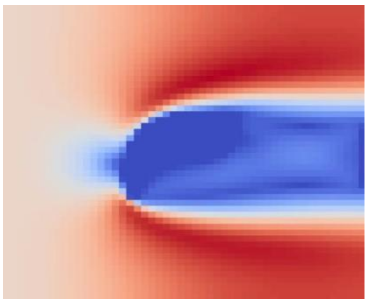
Table of Contents

Project support and dissemination of this compendium has been generously sponsored by:

HPC wire

Team 211

Deep Learning for Steady-State Fluid Flow Prediction in the Advania Data Centers Cloud



"The overhead of creating high volumes of samples can be effectively compensated by the high-performance containerized computing environment provided by UberCloud and Advania."

1 MEET THE TEAM

End-User: Jannik Zuern, Renumics GmbH, Karlsruhe, Germany
Software Provider: OpenFOAM open source CFD software
Resource Provider: Advania Data Centers Cloud, Iceland
HPC and AI Experts: Stefan Suwelack, Markus Stoll, and Jannik Zuern, Renumics; Joseph Pareti, AI Consultant; and Ender Guler, UberCloud Inc.

2 USE CASE

Solving fluid flow problems using Computational Fluid Dynamics (CFD) is demanding both in terms of computer power and in terms of simulation duration. Artificial neural networks (ANN) can learn complex dependencies between high-dimensional variables. This ability is exploited in a data-driven approach to CFD that is presented in this case study. An ANN is applied in predicting the fluid flow given only the shape of the object that is to be simulated. The goal of the approach is to apply an ANN to solve fluid flow problems to significantly decrease time-to-solution while preserving much of the accuracy of a traditional CFD solver. Creating a large number of simulation samples is paramount to let the neural network learn the dependencies between simulated design and flow field around it.

This project between Renumics GmbH and UberCloud Inc., which has been performed in 2018 for training and education purpose, explores the benefits of additional cloud computing resources that can be used to create a large amount of simulation samples in a fraction of the time a desktop computer would need to create them. In this project, we want to explore whether the overall accuracy of the neural network can be improved when more samples are being created in the UberCloud Container und then used during the training of the neural network. UberCloud kindly provided the cloud infrastructure, a CentOS Docker container with an OpenFOAM installation, and additional tech support during the project development.

3 WORKFLOW OVERVIEW

In order to create the simulation samples automatically, a comprehensive workflow was established.

As a **first step**, random two-dimensional shapes are created. These shapes have to be diverse enough to let the neural network learn the dependencies between different kinds of shapes and their respective surrounding flow fields.

In the **second step**, these shapes are meshed and added to an OpenFOAM simulation case template (Fig. 1). This template is simulated using the steady-state solver OpenFOAM solver simpleFOAM.

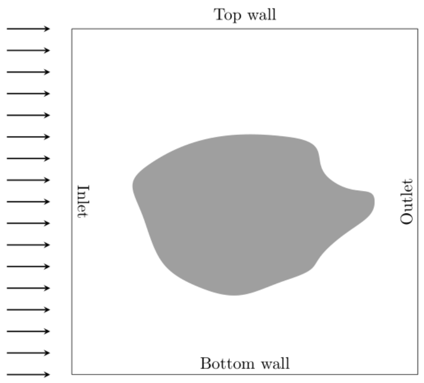


Figure 1: Simulation setup. The flow enters the simulation domain through the inlet, flows around the arbitrarily shaped obstacle (grey shade) and leaves the simulation domain through the outlet.

The **third step** post-processed the simulation results using open-source visualization ParaView. The flow-fields are resampled on a rectangular grid to simplify information processing by the neural net.

In the **fourth step** the simulated design and the flow fields are fed into the input queue of the neural network which, after training, is able to infer a flow field merely from seeing the to-be-simulated design.

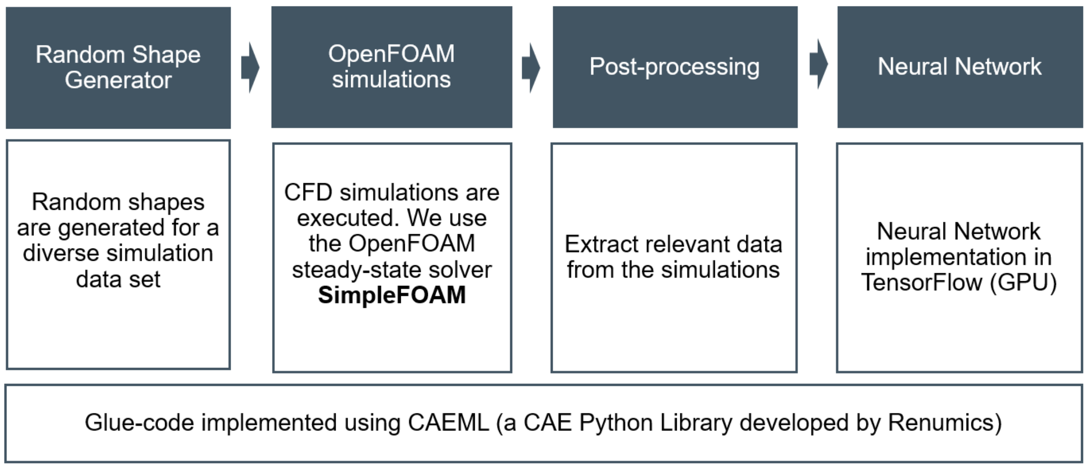

Random Shape Generator
OpenFOAM simulations
Post-processing
Neural Network
Random shapes are generated for a diverse simulation data set
CFD simulations are executed. We use the OpenFOAM steady-state solver SimpleFOAM
Extract relevant data from the simulations
Neural Network implementation in TensorFlow (GPU)
Glue-code implemented using CAEML (a CAE Python Library developed by Renumics)

Figure 2: Four-step Deep Learning workflow.

Hardware specs

The hardware specs of the Advania Data Centers compute node hosting UberCloud’s container are:

- 2 x 16 core Intel Xeon CPU E5-2683 v4 @ 2.10 GHz and 251 GB memory, no GPU

The hardware specs of the previously used desktop workstation are as follows:

- 2 x 6 core Intel i7-5820K CPU @ 3.30 GHz, and 32 GB memory
- GPU: GeForce GTX 1080 (8GB GDDR5X memory)

4 RESULTS

Time needed to create samples

As a first step, we compared the time it takes to create samples on the desktop workstation computer with the time it takes to create the same number of samples on the UberCloud container. Figure 3 illustrates the difference in time it took to create 10,000 samples. On the desktop computer it took 13h 10min to create these 10,000 samples. In the UberCloud OpenFOAM container in the Advania Data Centers Cloud, it took about 2h 4min to create 10,000 samples, which means that a speedup of 6.37 could be achieved using the UberCloud container.

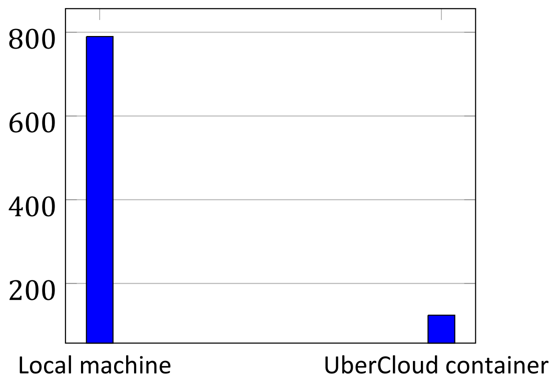
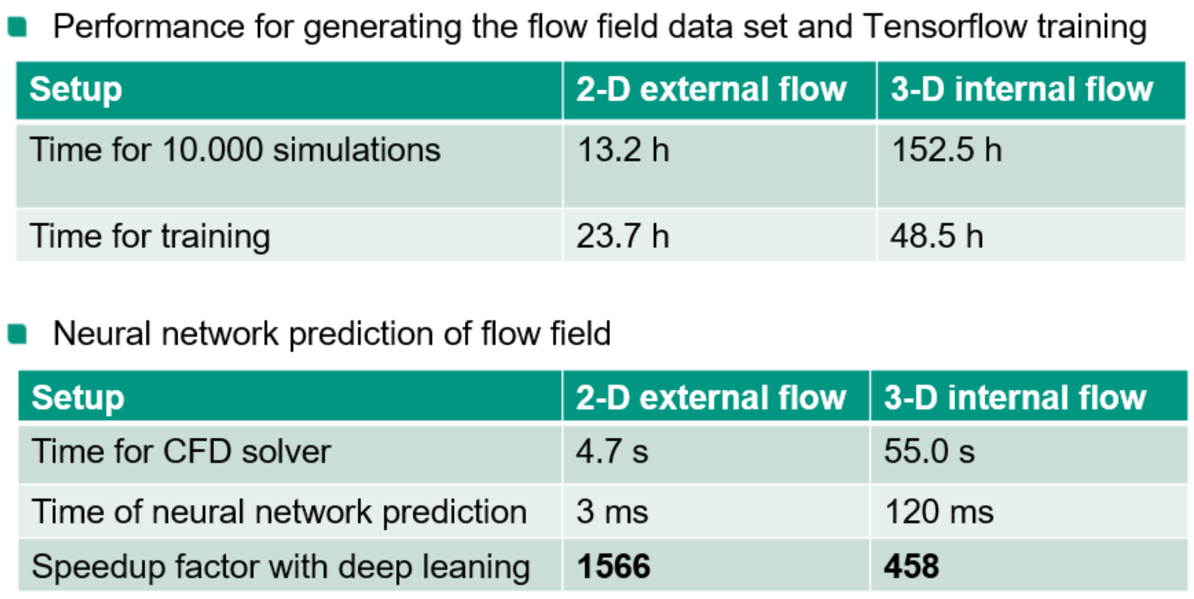


Figure 3: Comparison (in minutes) between Local machine and UberCloud container.

Neural Network performance evaluation

A total of 70,000 samples were created. We compare the losses and accuracies of the neural network for different training set sizes. In order to determine the loss and the accuracy of the neural network, we first must define, what these terms actually mean.



- Performance for generating the flow field data set and Tensorflow training

Setup	2-D external flow	3-D internal flow
Time for 10.000 simulations	13.2 h	152.5 h
Time for training	23.7 h	48.5 h

- Neural network prediction of flow field

Setup	2-D external flow	3-D internal flow
Time for CFD solver	4.7 s	55.0 s
Time of neural network prediction	3 ms	120 ms
Speedup factor with deep leaning	**1566**	**458**

Figure 4: Performance and speedup of flow simulations with neural network prediction.

Definitions

__Loss:__ The loss of the neural network prediction describes how wrong the prediction of the neural network was. The output, or prediction, of the neural network in our project is a $N \times M \times 2$ tensor since the network tries to predict a fluid flow field with N elements in x-direction, M elements in y-direction, and two flow velocity components (velocity in x-direction and velocity in y-direction). A mean-squared-error metric was used to calculate the loss *l*:

$$l = \frac{1}{2} \sum_{d=0}^{1} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (v_{dij} - \bar{v}_{dij})^2 \tag{1}$$

where v_{dij} denotes the ground-truth velocity component in dimension *d* at the grid coordinates (*i,j*), $\overline{v}_{dij}$ denotes the predicted velocity component at the same position and in the same dimension. The goal of every machine learning algorithm is to minimize the loss of the neural network using numerical optimization schemes such as Stochastic Gradient Descent. Thus, a loss of 0.0 for all samples would mean that every flow velocity field in the dataset is predicted perfectly.

Accuracy: In order to be able to make sensible statements about the validity of the prediction of the neural network, metrics have to be defined that describe the level of accuracy that the neural network achieves. In general, the accuracy of a neural network describes how accurate the prediction of the neural network was. While the loss of a neural network is the metric that is being minimized during training, a small prediction loss does not necessarily mean that the corresponding prediction is physically meaningful. In general, however, a small prediction loss usually corresponds with a high accuracy. Different measurements of how accurate the outputs of the neural network are needed to express the validity of the predictions. A highly accurate prediction should have high values for all formulated accuracy measurements and a low loss at the same time. These accuracies can have values between 0.0 and 1.0, where an accuracy of 0.0 indicates that the prediction of the neural network does not at all coincide with the ground truth flow metric that is examined, and an accuracy of 1.0 means that the prediction coincides perfectly with the ground truth flow metric. Bear in mind that a low loss does not necessary cause high accuracy and vice versa. However, the two measurements are usually correlated.

In this study, two different accuracies were evaluated: Divergence accuracy and Drag accuracy:

- **Divergence accuracy:** Numerical CFD solvers aim to find a solution to the continuity equation and the momentum equation. For an incompressible fluid, the continuity equation dictates that the divergence of the velocity vector field is zero for every point in the simulation domain. This follows the intuition that at no point in the simulation domain fluid is generated (divergence would be greater than zero) or ceases to exist (divergence would be smaller than zero). By design, the Finite Volume Method preserves this property of the fluid even in a discretized form. A data-driven approach should as well obey this rule.

- **Cell accuracy:** The number of correctly predicted grid cells in the two- or three-dimensional grid yields an intuitive metric for how well the neural network predicts fluid flow behavior. As the network will never be able to predict the fluid flow velocity down to the last digit of a floating-point number, the following approach is proposed: If the relative error between the network prediction and the actual flow velocity is smaller than 5%, the respective grid cell is declared as predicted correctly. The cell accuracy can be calculated by counting the number of correctly predicted grid cells and dividing the results by the total number of grid cells.

5 TRAINING RESULTS

The generated samples are divided into the training and validation datasets. The training- and validation loss for different numbers of training samples was evaluated. Concretely, the neural net was trained three times from scratch with 1,000, 10,000, and 70,000 training samples respectively.
The following training parameters were used for all neural network training runs:

- Batch size: 32
- Dropout rate: 0.5
- Learning rate: 5×10^{-4}

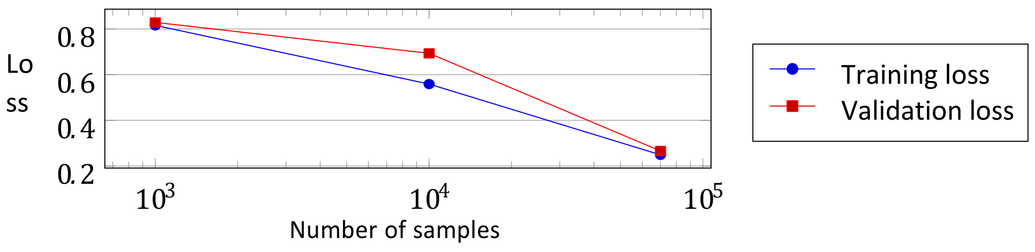


Figure 5: Loss after 50,000 training steps.

It can be observed that both training- and validation losses are lowest for the 70k samples training and are highest for the 1k training samples. The more different samples the neural network processes during the training process the better faster it is able to infer a flow velocity field from the shape of the simulated object suspended in the fluid. The validation loss tends to be higher than the training loss for all tested numbers of samples, which is a typical property of machine learning algorithms. Figure 6 shows the loss after 300,000 training steps:

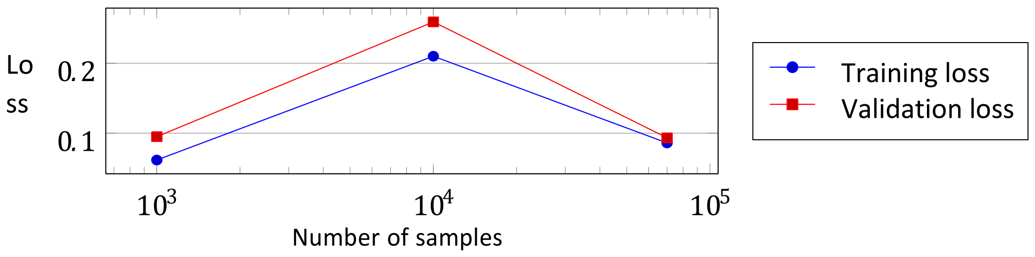


Figure 6: Loss after 300,000 training steps.

Surprisingly, the final training- and validation losses for the 70k samples training session are as low as the losses for the 1k samples training session. Generally speaking, no clear tendency towards lower losses when increasing the set of the training samples could be observed. This result is somewhat surprising since we expected the final losses at the end of the training process to show a similar tendency towards lower losses for higher numbers of samples. We assume that the number of samples does not heavily influence the final loss for extensive training sessions with many hundreds of thousand training steps. Finally, in Figure 7 the divergence and grid cell accuracy are visualized.

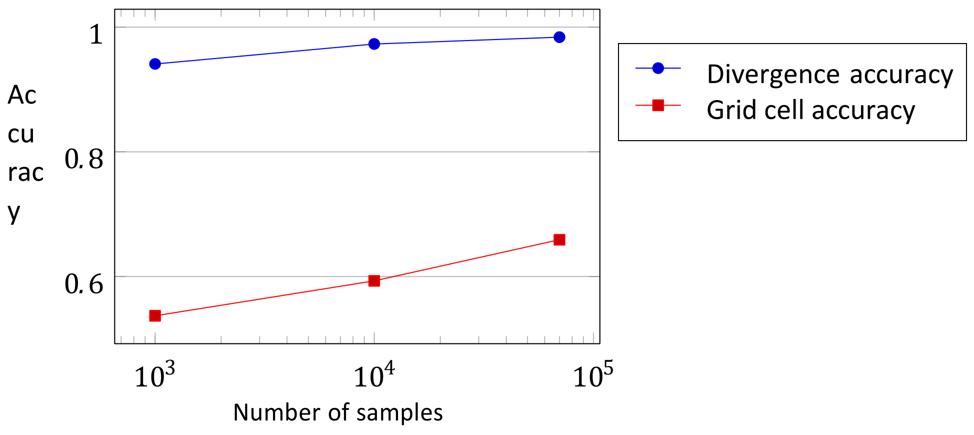


Figure 7: Validation accuracies after training.

Both the divergence accuracy and the grid cell accuracy show higher values for larger numbers of samples. While the divergence accuracy shows overall high values going from 0.94 for 1,000 samples to 0.98 for 70,000 samples, the grid cell accuracy also increases from a value of 0.53 for 1,000 samples to a value 0.66 for 70,000 samples. To recap: a grid accuracy of 0.66 means that approximately two thirds of all velocity grid cells were predicted correctly within 5% relative error to the correct value.

Figure 8 illustrates the difference between the ground truth flow field (left image) and the predicted flow field (right image) for one exemplary simulation sample after 300,000 training steps. The arrow direction indicates the flow direction and the arrow color indicates the flow velocity. Visually, no difference between the two flow fields can be made out.

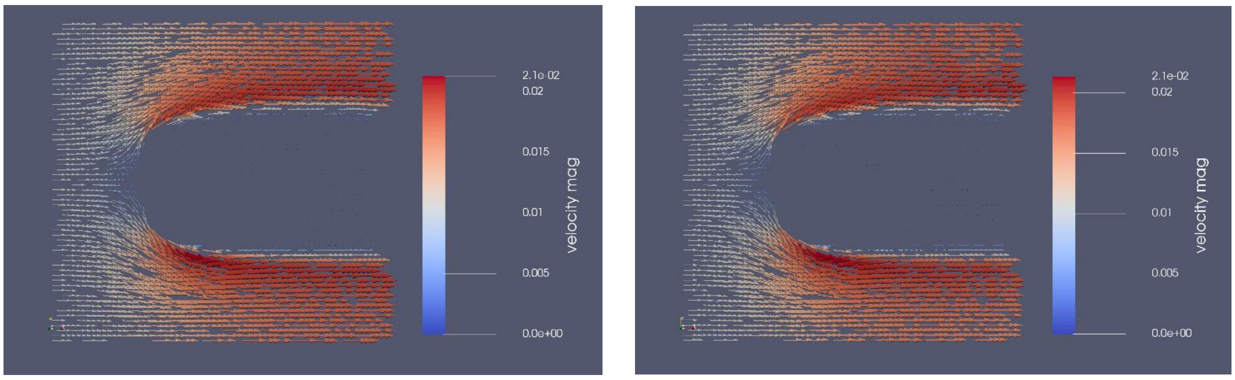


Figure 8: Exemplary simulated flow field (left image) and predicted flow field (right image).

CONCLUSION

We were able to prove a mantra amongst machine learning engineers: *The more data the better*. We showed that the training of the neural network is substantially faster using a large dataset of samples compared to smaller datasets of samples. Additionally, the proposed metrics for measuring the accuracies of the neural network predictions exhibited higher values for the larger numbers of samples. The overhead of creating high volumes of additional samples can be effectively compensated by the high-performance containerized (based on Docker) computing node provided by UberCloud on the Advania Data Centers Cloud. A speed-up of more than 6 compared to a state-of-the-art desktop workstation allows creating the tens of thousands of samples needed for the neural network training process in a matter of hours instead of days.

In order to train more complex models (e.g. for transient 3D flow models) much more training data will be required. Thus, software platforms for training data generation and management as well as flexible compute infrastructure will become increasingly important.

Published in 2018, Case Study Author – Jannik Zuern, Renumics GmbH

Team 212
Demonstrating a Machine Learning Model for Predictive Maintenance on Microsoft Azure

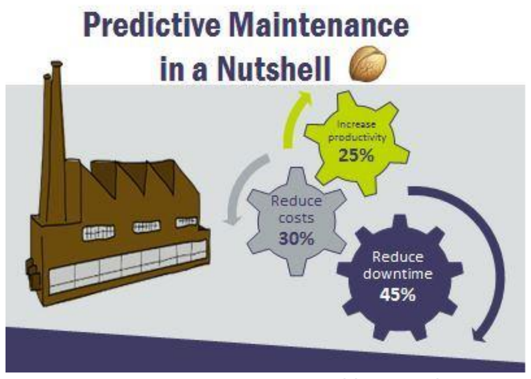


Figure 1: Infographics: https://goo.gl/cv7vLp

"This Machine Learning study is for predictive maintenance demonstration and learning purposes, and may be used as a baseline for the reader's custom applications."

MEET THE TEAM

End-User: Joseph Pareti, Artificial Intelligence Consultant
Software Provider: Open source Predictive Maintenance template provided by the Microsoft Azure Team
Resource Provider: Microsoft Azure Cloud
UberCloud Support: Wolfgang Gentzsch, President, The UberCloud
Microsoft Support: Yassine Khelifi, Escalation Engineer at Microsoft.

INTRODUCTION: HOW TO USE THIS DOCUMENT

The main purpose of this experimental work from 2019 is to create awareness on what is possible with machine learning in the manufacturing industry, and hence we picked a predictive maintenance use case. Next, there are 2 classes of users:

1. ***Developers need to follow through all steps described in the next paragraphs.*** You will do data ingestion, features engineering and train a Machine Learning model that can then be used to predict a machine likelihood of failure within a specified time window. If you are a developer, you need to work **all** notebooks. You will also need a front-end system (like a Windows PC running the Azure Machine Learning Workbench) and a back-end system such as a Data Science Virtual Machine in the Azure Cloud.
2. ***End-users who want to just consume the application as a service*** will need the assets files and a Docker image running on e.g. an Ubuntu server. Please send a request for the assets files to joepareti54@gmail.com.

USE CASE

The Predictive Maintenance model described in this report is open source and can be applied to different equipment types for which telemetry data and maintenance data records are available. A demo version is described in the following paragraphs and it is based on specific prerequisites. Customization of the model to real customers' case[1] is out of scope for this report but can be done on a project basis, explained in the conclusion/ recommendations paragraph.

The demo version is based on the following assumptions: 4 machine types are considered, each machine has 4 components, and there is *telemetry* data available on voltage, vibration, speed, etc. as well as maintenance records (indicating when components were replaced on what machine), error logs (not necessarily implying failure), machine characteristics, and how long each machine has been in service. The model is built in 4 stages each of which is a Jupyter Notebook:

1. Data ingestion
2. Feature engineering
3. ML model
4. Operationalization

Notebook 1, Data ingestion, is about accessing datasets and converting data into py-spark data-frames that are stored in Azure storage container (AKA blob storage), so that the data is accessed in the next notebook. The demo version uses pandas as source data from a SQL server in github, but it can obviously be replaced with customer's data.

Notebook 2, Feature engineering, loads the data sets created in Notebook 2 from an Azure storage container and combines them to create a single data set of features (variables) that can be used to infer a machine health condition over time. The notebook steps through several feature engineering and labeling methods to create this data set for use in the predictive maintenance machine learning solution.
The goal is to generate a single record for each time unit within each asset. The record combines features and labels to be fed into the machine learning algorithm.
Predictive maintenance takes historical data, marked with a timestamp, to predict current health of a component and the likelihood of failure within some future window of time. These problems can be characterized as a *classification method* involving *time series* data. Time series, since we want to use historical observations to predict what will happen in the future. Classification, because we classify the future as having a likelihood of failure.

Notebook 3, The ML model, uses the labeled feature data set constructed in Notebook 2, it loads the data from the Azure Blob container and splits it into a training and test data set. We then build a machine learning model (a decision tree classifier or a random forest classifier) to predict when different components within our machine population will fail. Two different classification model approaches are available in this notebook:

- **Decision Tree Classifier**: Decision trees and their ensembles are popular methods for the machine learning tasks of classification and regression. Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the

[1] Customization may mean working with different datasets (this is the easiest step), or changing the data structure and ML model according to customers' needs.

multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.

- **Random Forest Classifier**: A random forest is an ensemble of decision trees. Random forests combine many decision trees in order to reduce the risk of overfitting. Tree ensemble algorithms such as random forests and boosting are among the top performers for classification and regression tasks.

Notebook 4, Operationalization is about deploying the model into a Docker container for users to consume it as-a-web service. We propose an Ubuntu implementation.

Finally, some test cases have been run, and the relevance of input data on machine failures is shown.

SYSTEM ARCHITECTURE

The Predictive Maintenance application consists of three components**:**

1. A front-end application, Azure Machine Learning Workbench (AML), that runs on a local system, in my case a Windows 10 laptop

2. A back-end system, a Data Science Virtual Machine (DSVM) in the Azure cloud. The main advantage of working with a DSVM is that all required software components for ML are pre-installed and maintained by Microsoft. Moreover, you can configure the DSVM to fulfill your needs, in my case I selected the minimum required configuration, minimum cost, in terms of number of vCPUs, memory, local storage since the exercise was not intended to measure any performance data

3. A blob-storage account (in Azure) to host the intermediate data that are transferred from one notebook to the other.

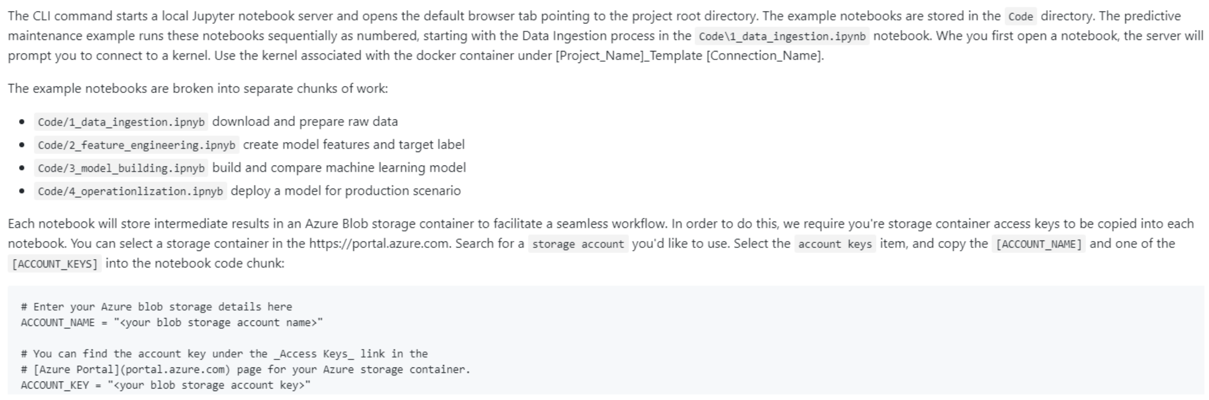
The front-end and back-end applications interact over the internet; a stable standard configuration Wireless LAN has proven to be adequate for the job. If your network is unstable, you may face some issues that are described in the Appendix of this report, available upon request from wgentzsch@gmail.com.

DEVELOPER VIEW

If you are a developer, you need to go through all steps starting with data ingestion, next you do feature engineering, and finally you train the model. The basic tool that will guide you all along is the Jupyter Notebook that will be started in AML.

You create a new project in AML, and then select the *General Predictive Maintenance* scenario[2]; AML comes with on-line documentation to guide you through the steps. You need to first ensure that the DSVM is up and running so that the front-end can attach to it via remote Docker. On the front-end, you need a CLI: you can use power shell (which I recommend), or the DOS command interface to do the basic startup commands:

[2] Microsoft provided 2 Predictive Maintenance templates: the other one is Deep Learning for PdM which is however oversimplified.



The CLI command starts a local Jupyter notebook server and opens the default browser tab pointing to the project root directory. The example notebooks are stored in the `Code` directory. The predictive maintenance example runs these notebooks sequentially as numbered, starting with the Data Ingestion process in the `Code\1_data_ingestion.ipynb` notebook. Whe you first open a notebook, the server will prompt you to connect to a kernel. Use the kernel associated with the docker container under [Project_Name]_Template [Connection_Name].

The example notebooks are broken into separate chunks of work:

- `Code/1_data_ingestion.ipynb` download and prepare raw data
- `Code/2_feature_engineering.ipynb` create model features and target label
- `Code/3_model_building.ipynb` build and compare machine learning model
- `Code/4_operationlization.ipynb` deploy a model for production scenario

Each notebook will store intermediate results in an Azure Blob storage container to facilitate a seamless workflow. In order to do this, we require you're storage container access keys to be copied into each notebook. You can select a storage container in the https://portal.azure.com. Search for a `storage account` you'd like to use. Select the `account keys` item, and copy the `[ACCOUNT_NAME]` and one of the `[ACCOUNT_KEYS]` into the notebook code chunk:

```
# Enter your Azure blob storage details here
ACCOUNT_NAME = "<your blob storage account name>"

# You can find the account key under the _Access Keys_ link in the
# [Azure Portal](portal.azure.com) page for your Azure storage container.
ACCOUNT_KEY = "<your blob storage account key>"
```

Figure 1: Preparation steps on the AML front-end application (please enlarge your document).

RESULTS

The result of this study is a demo version that runs in a Docker container. Please contact me (joepareti54@gmail.com) if you want to test it yourself. In addition, I can provide workshops on AI applications for CAE and manufacturing.

PERFORMANCE BENCHMARKING

This is out-of-scope for the current project. On my DSVM, the by far most time-consuming part of the application is Notebook 2 (feature engineering), which takes 71 minutes to complete: table joins and label construction run on all 4 vCPUs in the DSVM. Notebook 3, model training, takes approximately 10 minutes. If you plan to use a larger data-set we will need to size the system, preferably with GPUs.

BENEFITS

In general, the benefits of avoiding down time of costly equipment, avoiding loss of business due to poor response time, or damage to a provider's reputation due to service outages, are self-explanatory. Machine Learning is an effective technology to accomplish those goals. According to Forbes[3], improving preventative maintenance and Maintenance, Repair and Overhaul (MRO) performance with greater predictive accuracy to the component and part-level is *one of the 10 ways ML is revolutionizing manufacturing.*

There are several implementations of Predictive Maintenance (PdM) using ML techniques out there. The following lists some PdM applications I came to know in a short space of time:

1. Rapidminer
2. Mathworks for Predictive Maintenance
    - Toolbox for tasks such as RUL estimation, condition parameters design, statistical methods, ML, features extraction from data, label construction for supervised training, etc.
    - *Toolbox* means the user must build the application
    - Use cases for rolling bearings, gearbox, pumps, etc.
    - Ability to simulate failure data using number crunching code from Mathworks

[3] https://www.forbes.com/sites/louiscolumbus/2016/06/26/10-ways-machine-learning-is-revolutionizing-manufacturing/#12fa1af328c2

3. Vargroup
4. Microsoft
5. General Electric

The benefits of working with the PdM model presented in this report instead of starting from scratch are:

(i) open source code,
(ii) extensive debugging and testing on the demo version, and
(iii) a core team of experts has been identified.

CONCLUSION & RECOMMENDATIONS

The subject matter of this report is an open source implementation of PdM, as such customers have free access to the code. Users are encouraged to approach us, and test the demo version "as-is" to evaluate how well it models their environment.

Next, we could work in a team (including the customer, UberCloud, myself, Microsoft and Nvidia) to scope the project and create a Statement of Work (SOW): this will also specify what data, data structures and interfaces are needed. In some cases, it may be possible to just acquire and integrate customer's data[4]. In the SOW, we will also investigate what ML models are more suitable to the customer's use case. The next step will be a proof of concept prior to production-ready applications.

FINAL UPDATE

Microsoft has released an updated version of its predictive maintenance demo version as part of their GitHub open source samples initiative: this version promises the following new features:

- Training and operationalization using Azure Machine Learning Service
- Model training on Azure Databricks cluster (gaining 1 order of magnitude in speed)
- Prediction Serving using Azure container instance
- Minor changes include:
  - Replacement of pandas data frame manipulation to spark data
  - Addition of a features (variables) importance plot
  - Slight modification to existing Exploratory Data Analysis (EDA) plots.

While our plan going forward calls for testing this new version, at this time we cannot publish any usage experience on it. If you – the reader – are interested in a joint exploration of this new version please contact us.

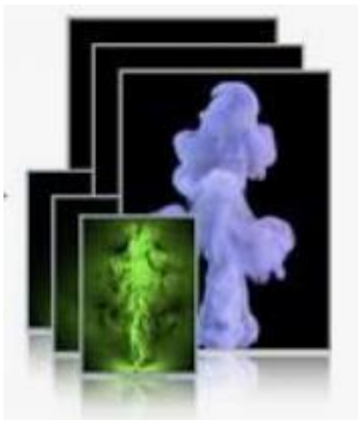
From an initial look at the new repository, it appears that the functional parts of the application are aligned with what we have already reported in this case study.

Performed in 2019, Case Study Authors – Joseph Pareti and Yassine Khelifi

[4] Some advice here: https://gallery.azure.ai/Collection/Predictive-Maintenance-Modelling-Guide-1

Team 213

Deep Learning in Computational Fluid Dynamics in the Microsoft Azure Cloud


Courtesy, Thurey Group, TUM

"Applying Deep Learning to highly complex compute intensive CFD by creating a database of synthetic flow descriptors that correlate with a given complex problem, so that the accurate solution could be inferred by a coarse-grid and much less compute intensive solution is one of the great benefits of this method."

MEET THE TEAM
End-User: Joseph Pareti, Artificial Intelligence Consultant
Software Provider: Nils Thurey and Mengyu Chu, Technical University of Munich, Germany
Resource Provider: Microsoft Azure Cloud
UberCloud Support: Wolfgang Gentzsch, President, The UberCloud
Microsoft Support: Yassine Khelifi, Escalation Engineer at Microsoft.

INTRODUCTION: USING ARTIFICIAL INTELLIGENCE TO SPEED UP A CFD APPLICATION
In this work, performed in 2020, Convolution Neural Networks (CNN) are used for computing feature descriptors for density and velocity fields in smoke clouds. The CNN learns from a repository of computed results using the MANTAFLOW application. The CNN training, using TensorFlow, determines flow descriptors for density and velocities and a flow similarity score. In addition, the model includes a deformation limiting patch advection with anticipation module which enhances the stability and performance.

When given a coarse simulation, the model associates with a high degree of confidence a more accurate simulation that is retrieved from a database of computed results. When compared with a traditional approach, the CNN-based approach provides much faster time to solution with comparable accuracy as when using a sufficiently fine grid.

There is an interest among CAE users to accelerate the time-to-solution of numerically intensive applications, in order to run a sufficient number of cases for product optimization using a parametric approach. Hence it is expected that more applications will become available that use deep networks to either replace compute intensive modules, such as FluidNet, or replace the entire computation as described. Another reason to select this application for an UberCloud case study was the interest by the developer at TUM to contribute to our work in the form of Q&A, troubleshooting, etc.

Finally, we've made use of Microsoft Azure support for questions related to the configuration of the Data Science Virtual Machine (DSVM). If you are interested to run the application and you need more details than explained in this report, please send us a request at joepareti54@gmail.com.

USE CASE

Predicting smoke flows is a common task in computer graphics using CFD models. This can be done in 2D or 3D. Because an accurate calculation of flow properties is time consuming, researchers at the TUM developed new algorithms that benefit from deep learning technologies to dramatically reduce time-to-solution. While this paper is about a specific flow problem, with a specific solver (MANTAFLOW), it should be regarded as a feasibility study, and hence we encourage general purpose CFD users to look *beyond the box*, and come back to us with their specific CFD (or CAE) use case that could also benefit from deep learning or machine learning approaches.

At TUM, Nils Thurey and Mengyu Chu, in their work about Data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors, take the following perspective to efficiently realize high-resolution flows (see Figure 1): they propose to use a fluid repository, consisting of a large collection of pre-computed space-time regions. From this, they synthesize new high-resolution volumes. In order to very efficiently find the best match from this repository, they propose to use novel, flow-aware feature descriptor. they ensure that L2 distances in this feature space will correspond to real matches of flow regions in terms of fluid density as well as fluid motion, so that they can very efficiently retrieve entries even for huge libraries of flow datasets.

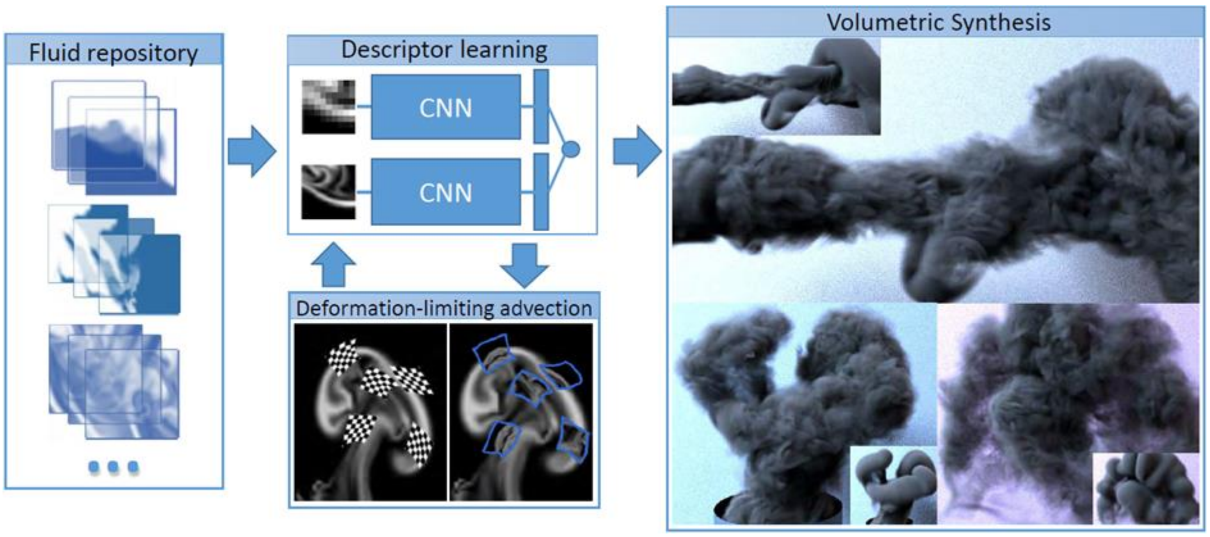


Figure 1: Realizing high-resolution flows efficiently by using a fluid repository, consisting of a large collection of pre-computed space-time regions.

SYSTEM ARCHITECTURE

In this application, we first ***built the MANTAFLOW application*** to calculate the exact data that will be used to train the deep network. There are 2D and 3D datasets. Next, we ***ran MANTAFLOW***. The output data of MANTAFLOW will be fed in ***TensorFlow*** to train the deep network. Once the network is trained we were able ***to make inferences*** and compare “exact” results (i.e. from MANTAFLOW with a fine grid) and results from the deep network. For the interested reader, all steps listed above are explained in detail in the Appendix available from wgentzsch@gmail.com. The paragraph on Results shows the exact and approximated output for the *smoke_tiled* example.

RESULTS

We achieved the following results:

1. An Azure-based demo version of the application that uses deep learning to significantly reduce time-to-solution in a CFD application.
2. A demo version that was employed for a specific example called the *smoke_tiled problem.*

In Figure 2 we compare the exact solution (right) with the approximate solution (left) for 1 sample:

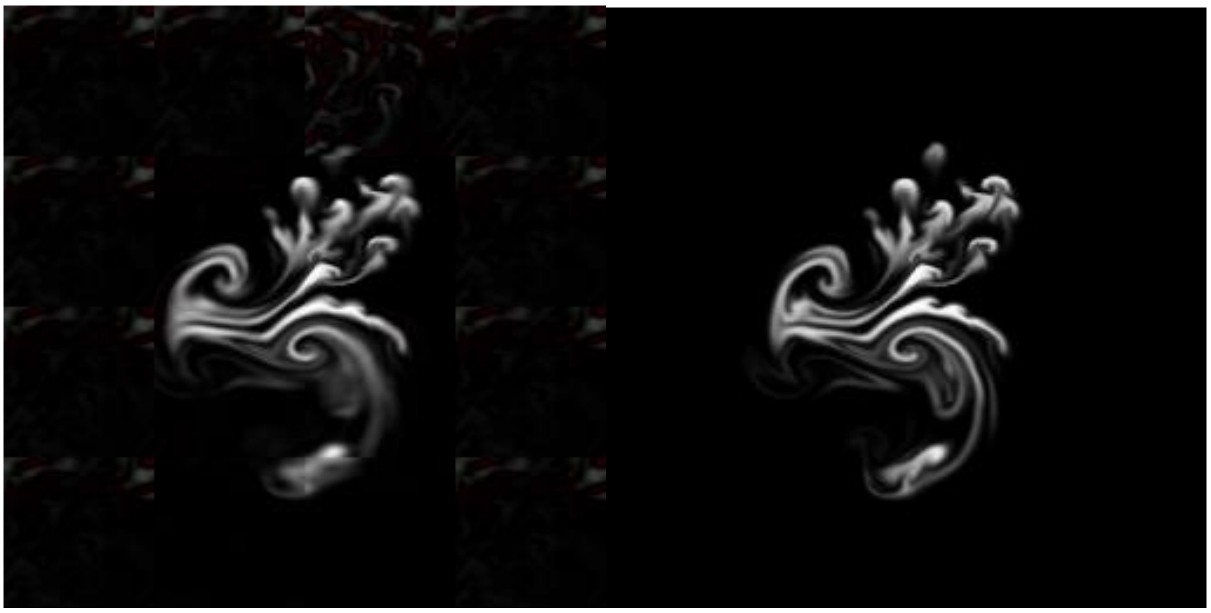


Figure 2: Smoke flow approximated by deep learning (left) vs. accurate MANTAFLOW calculation (right).

PERFORMANCE BENCHMARKING

This is out-of-scope for the current project. On our DSVM, the test cases presented in this report run in a few minutes; the exercise was just intended to validate the application and not to record any performance data.

BENEFITS

There are several R&D organizations working on deep learning tools to accelerate time-to-solution of CAE applications. In CFD, the deep learning approach varies:

1. One could aim at creating a database of synthetic flow descriptors that correlate with a given problem, so that the accurate solution could be inferred by a coarse grid solution (which is obviously less compute intensive). This is the approach taken by TUM in this report.
2. Another way to exploit the power of deep learning is by replacing parts of the computation by means of a deep network that simulates the exact behavior; this is the case of FluidNet where the non-linear Navier-Stokes partial differential equations are solved numerically, while the Poisson pressure correction term is simulated by a deep network instead of being calculated using traditional algorithms like sparse matrix solvers.
3. Or one could accurately calculate a number of cases, and then train the network using those results so that the network can predict the fluid flow based on geometry alone; this is the approach taken by Renumics in a recent UberCloud case study.[5]

If you are a CAE engineer, you may want to follow the methodology presented in this report and then possibly adapt it to your real problem, or work with us to design a deep learning-based solution.

CONCLUSION & RECOMMENDATIONS

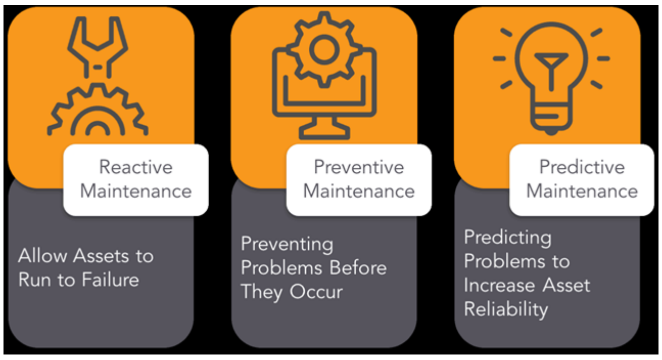
[5] Team 211: Deep Learning for Steady-State Fluid Flow Prediction in the Advania Data Centers Cloud

The subject matter of this report is a demonstration of how engineering simulations like computational fluid dynamics can benefit from Deep Learning with TensorFlow. Engineers are encouraged to approach us, get inspired for their work, and test this demo version "as-is", to evaluate how their own environment can benefit from this or a similar approach.

As a next step for interested readers, we suggest working together in a team (including the customer, Joseph Pareti, UberCloud, Microsoft, and Nvidia) to identify an internal application, scope the project, and create a Statement of Work (SOW): this will specify what algorithm can be used that takes advantage of Deep Learning / Machine Learning. The next step will be a Proof of Concept prior to production ready applications.

Project performed in 2020, Case Study Author – Joseph Pareti and Wolfgang Gentzsch

Team 214
Demonstrating a Machine Learning Model for Predictive Maintenance on Microsoft Azure Part II *


Reactive Maintenance
Allow Assets to Run to Failure
Preventive Maintenance
Preventing Problems Before They Occur
Predictive Maintenance
Predicting Problems to Increase Asset Reliability

Picture, Courtesy: VIZIYA Corporation

"This Machine Learning study (update from Team 212) is for predictive maintenance demonstration and learning purposes, and may be used as a baseline for the reader's own custom application."

MEET THE TEAM

End-User: Joseph Pareti, Artificial Intelligence Consultant
Software Provider: Open source Predictive Maintenance template by the Microsoft Azure Team
Resource Provider: Microsoft Azure Cloud
UberCloud Support: Wolfgang Gentzsch, President, The UberCloud
Microsoft Support: Yassine Khelifi, Escalation Engineer at Microsoft.

USE CASE

The Predictive Maintenance model described in this report from 2020 is open source and can be applied to different equipment types for which telemetry data and maintenance data records are available. **The implementation described herein assumes an Azure cloud subscription and some operating knowledge on Azure.**

Four machine types are considered, each machine has four components, and there is *telemetry* data available on voltage, vibration, speed, and pressure, as well as maintenance records (indicating when last a component was replaced on what machine), error logs (not necessarily implying failure), machine characteristics, and how long each machine has been in service. The model is built in four stages each of which is implemented in a Jupyter notebook running Python version3:

1. Data ingestion
2. Feature engineering
3. ML model
4. Operationalization

*) Note: This document is the continuation of our case study 212, about a recent release of the predictive maintenance software by Microsoft. The functionality of the model, and predictive precision and accuracy are the same as in the first release, however the current implementation is based on Azure Machine Learning Services and databricks that significantly reduces time-to-solution, while simplifying the end-user's administration task, since the entire application runs in Azure. Therefore, the desk-top front end, "Azure Machine Learning Workbench" which was a required component in the first release, has been removed.

Notebook 1: Data ingestion is about accessing the datasets from blob storage, cleaning the data, and storing the data as a SPARK dataframe in cluster for further processing by the next notebooks.

Notebook 2: Feature engineering loads the data sets created in the **Data Ingestion** notebook and combines them to create a single data set of features (variables) that can be used to infer a machine health condition over time.
The goal is to generate a single record for each time unit within each asset. The record includes features and labels to be fed into the machine learning algorithm.

Predictive maintenance takes historical data, marked with a timestamp, to predict current health of a component and the probability of failure within some future window of time. These problems can be characterized as a *classification method* involving *time series* data. Time series, since we want to use historical observations to predict what will happen in the future. Classification, because we classify the future as having a probability of failure.

Note that this step requires significant data science and programming skills, however it is separate from classical Machine Learning tasks. It also requires domain expertise in order to focus on relevant features for the task at hand.

Notebook 3: The ML model uses the labeled feature data set constructed in notebook 2, it loads the data and splits it into a training and test data set. We then build a machine learning model (a decision tree classifier or a random forest classifier) to predict when different components within our machine population will fail.

Two different classification model approaches are available in this notebook:

- **Decision Tree Classifier**: Decision trees and their ensembles are popular methods for the machine learning tasks of classification and regression. Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.
- **Random Forest Classifier**: A random forest is an ensemble of decision trees. Random forests combine many decision trees in order to reduce the risk of overfitting. Tree ensemble algorithms such as random forests and boosting are among the top performers for classification and regression tasks.

Notebook 4: Operationalization, loading the model from the Code/3_model_building.ipynb Jupyter notebook and the labeled feature data set constructed in the Code/2_feature_engineering.ipynb notebook in order to build the model deployment artifacts. The notebook is used to deploy and operationalize the model and is built on the Azure Machine Learning service SDK.

An appendix at the end of this report (available from wgentzsch@gmail.com) provides additional detail on the 4 notebooks. **If you wish a complete input/output set for this use case, e.g. to compare with your own work, please send e-mail to joepareti54@gmail.com.**

AZURE MACHINE LEARNING SDK ON AZURE DATABRICKS

The ML SDK and databricks are options to implement custom AI which require custom data and model training. The Azure ML SDK defines a workspace that contains compute and storage resources, as well as models, experiments (i.e. all attempts with different parameters), and deployment services such as Docker images.

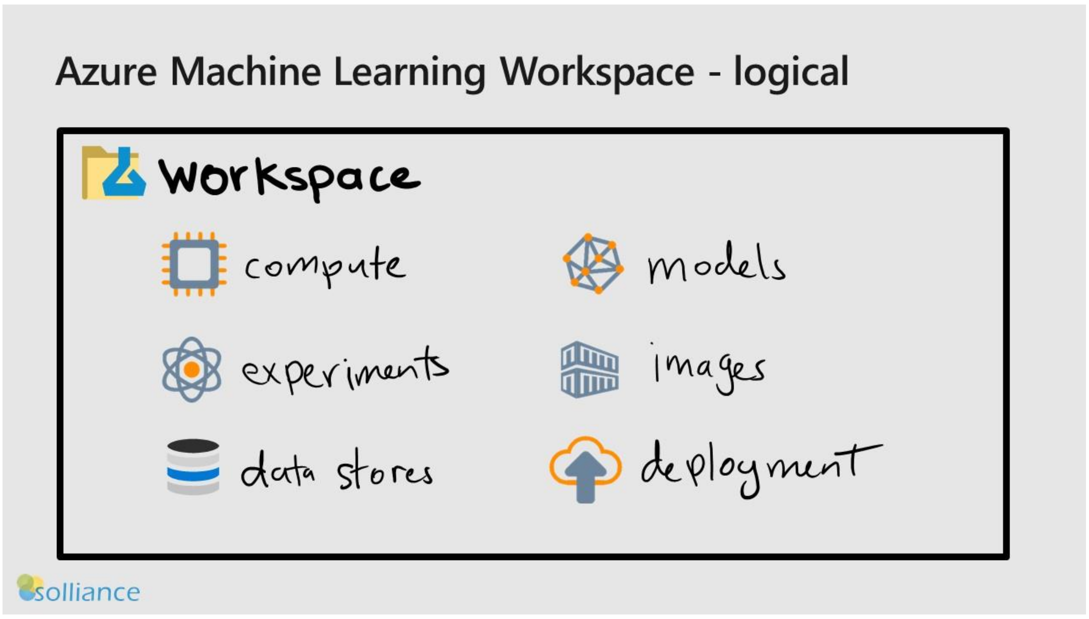


Figure 1: Azure Machine Learning Workspace.

Databricks is an architecture for Spark environments that provides out-of-the-box support for common interfaces and supports at-scale Machine Learning deployments:

- Azure Databricks is a fully-managed, cloud-based Big Data and Machine Learning platform.
- It empowers developers to accelerate AI and innovation by simplifying the process of building enterprise-grade production data applications
- It is built on a joint effort by the team that started Apache Spark and Microsoft
- It is a single platform for big data processing and Machine Learning.

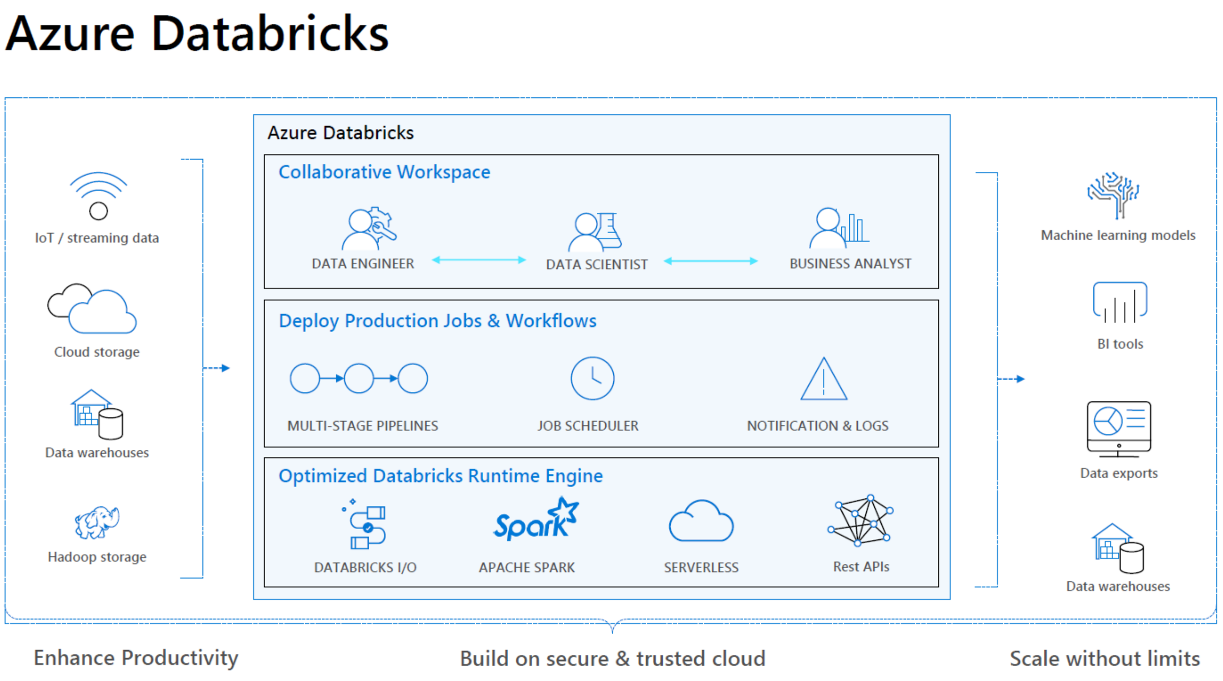


Figure 2: Azure Databricks.

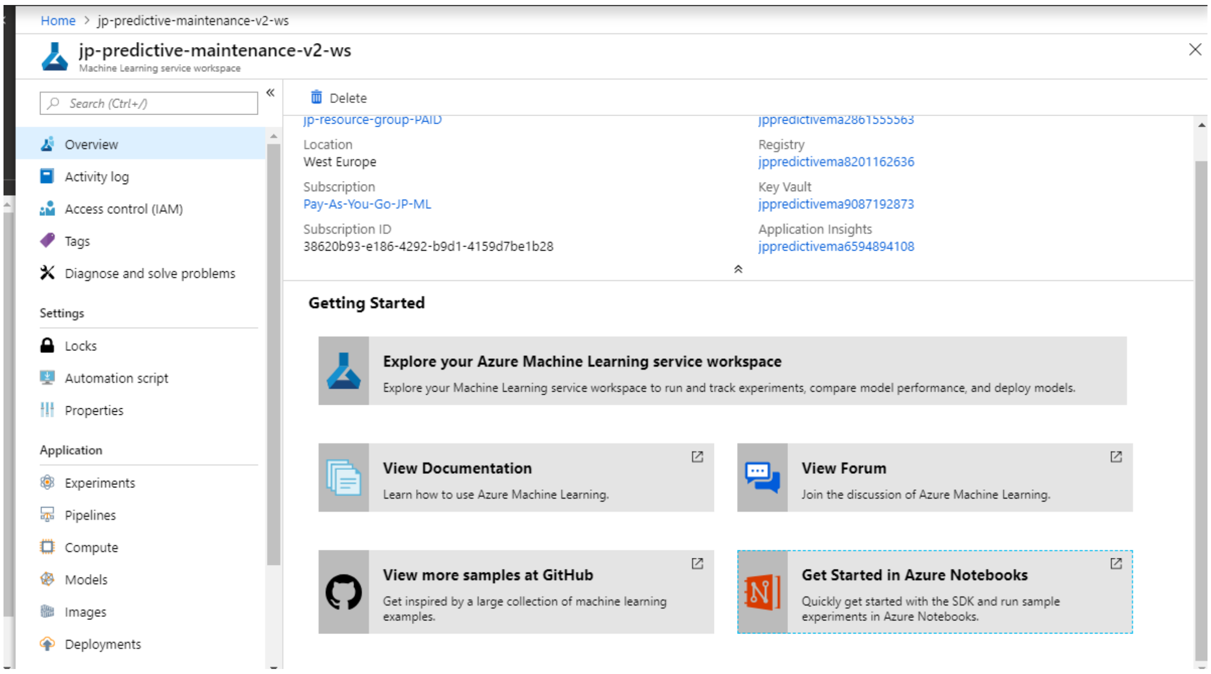
SYSTEM ARCHITECTURE USED FOR THIS CASE STUDY

The following is a list of the software components that must be implemented:

- A workspace in Azure
- A development environment for ML
- An Azure Databricks cluster deployed with the following configuration:
    - Databricks Runtime version: (latest stable release) (Scala 2.11)
    - Python version: 3
    - Driver/Worker type: Standard_DS13_v2
    - Python libraries installed:

      ipython==2.2.0, pyOpenSSL==16.0.0, psutil, azureml-sdk[databricks], cryptography==1.5

In the implementation for this case study, the Databricks cluster includes 2-8 nodes that are DS3_v2 virtual machines, which is a low-cost option to demonstrate the feasibility of the application. screen dumps below provide additional detail on utilized software components.



jp-resource-group-PAID
Location
West Europe
Subscription
Pay-As-You-Go-JP-ML
Subscription ID
38620b93-e186-4292-b9d1-4159d7be1b28

jppredictivema2861555563
Registry
jppredictivema8201162636
Key Vault
jppredictivema9087192873
Application Insights
jppredictivema6594894108

Getting Started

Explore your Azure Machine Learning service workspace
Explore your Machine Learning service workspace to run and track experiments, compare model performance, and deploy models.

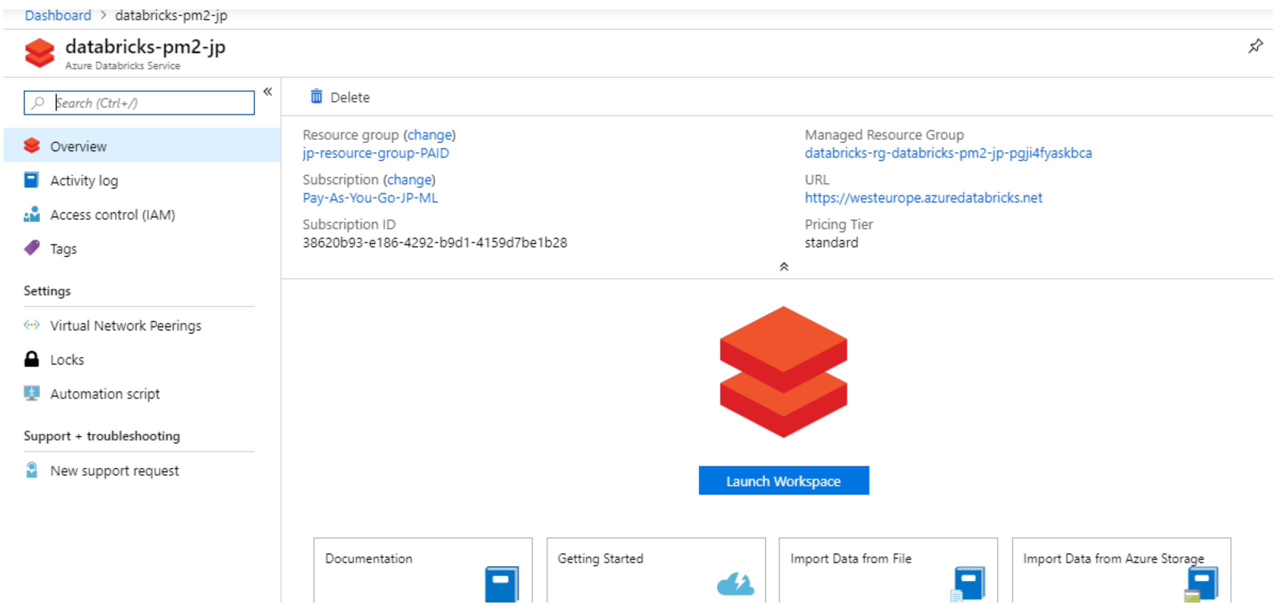
View Documentation
Learn how to use Azure Machine Learning.

View Forum
Join the discussion of Azure Machine Learning.

View more samples at GitHub
Get inspired by a large collection of machine learning examples.

Get Started in Azure Notebooks
Quickly get started with the SDK and run sample experiments in Azure Notebooks.

Figure 3: Azure Machine Learning Workspace; use the "view forum" option (bottom right) to ask Microsoft product managers for advice.



Resource group (change)
jp-resource-group-PAID

Subscription (change)
Pay-As-You-Go-JP-ML

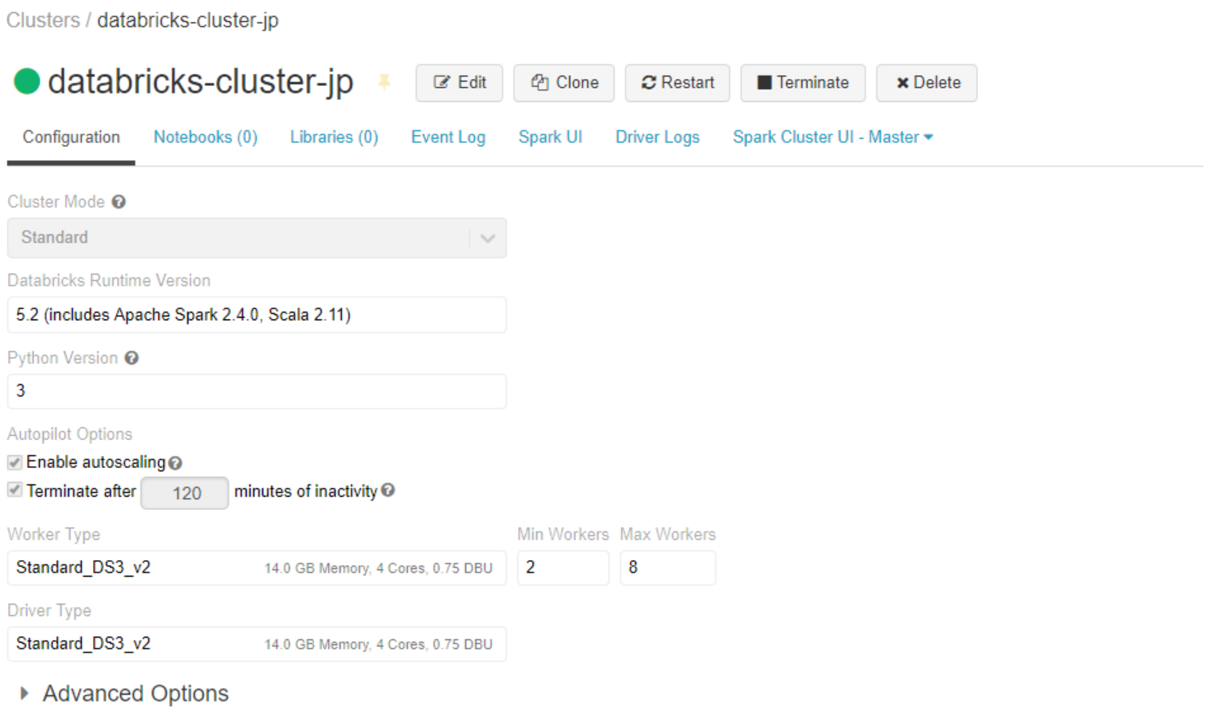
Subscription ID
38620b93-e186-4292-b9d1-4159d7be1b28

Managed Resource Group
databricks-rg-databricks-pm2-jp-pgji4fyaskbca

URL
https://westeurope.azuredatabricks.net

Pricing Tier
standard

Figure 4: Databricks Workspace.



Cluster Mode: Standard

Databricks Runtime Version: 5.2 (includes Apache Spark 2.4.0, Scala 2.11)

Python Version: 3

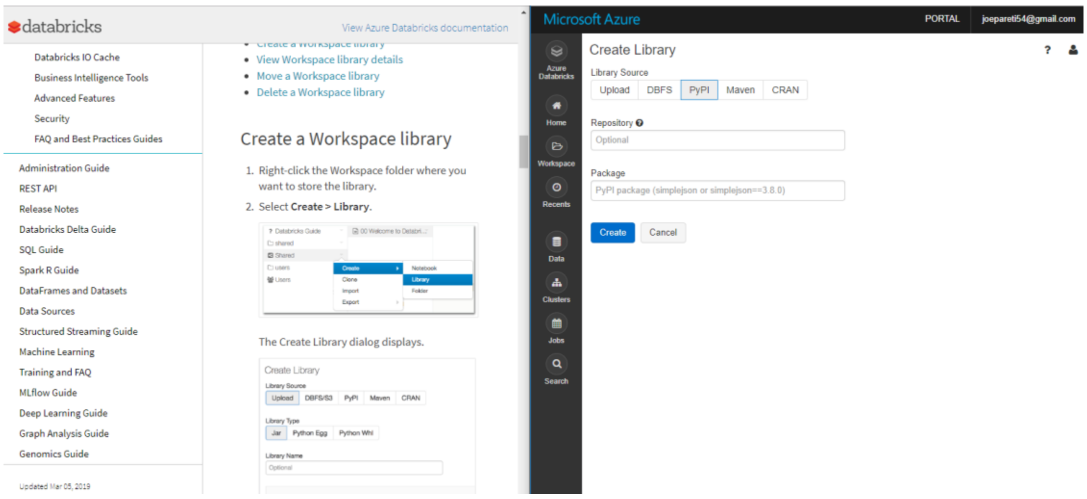
Autopilot Options
- ✓ Enable autoscaling
- ✓ Terminate after 120 minutes of inactivity

Worker Type: Standard_DS3_v2 14.0 GB Memory, 4 Cores, 0.75 DBU

Min Workers: 2 Max Workers: 8

Driver Type: Standard_DS3_v2 14.0 GB Memory, 4 Cores, 0.75 DBU

Figure 5: Databricks cluster; set the terminate flag to avoid being charged after the job is done; enable autoscaling to allow databricks to grow or shrink according to job requirement.



- View Workspace library details
- Move a Workspace library
- Delete a Workspace library

Create a Workspace library

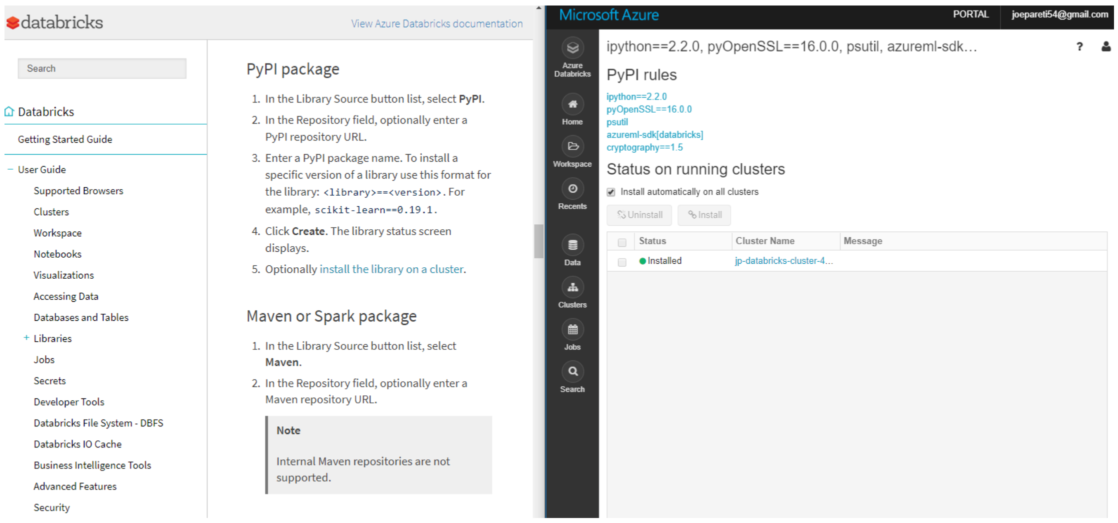
1. Right-click the Workspace folder where you want to store the library.
2. Select **Create > Library**.

The Create Library dialog displays.

Create Library
Library Source: [✓ Upload] DBFS/S3 PyPI Maven CRAN
Library Type: [✓ Jar] Python Egg Python Whl
Library Name: Optional

Create Library
Library Source: Upload DBFS [✓ PyPI] Maven CRAN
Repository: Optional
Package: PyPI package (simplejson or simplejson==3.8.0)
Create Cancel

Figure 6: Libraries on databricks cluster. Left: excerpt from the User's guide, Right: configuration used for this case study.



PyPI package

1. In the Library Source button list, select **PyPI**.
2. In the Repository field, optionally enter a PyPI repository URL.
3. Enter a PyPI package name. To install a specific version of a library use this format for the library: `<library>==<version>`. For example, `scikit-learn==0.19.1`.
4. Click **Create**. The library status screen displays.
5. Optionally install the library on a cluster.

Maven or Spark package

1. In the Library Source button list, select **Maven**.
2. In the Repository field, optionally enter a Maven repository URL.

Note

Internal Maven repositories are not supported.

ipython==2.2.0, pyOpenSSL==16.0.0, psutil, azureml-sdk…

PyPI rules

ipython==2.2.0
pyOpenSSL==16.0.0
psutil
azureml-sdk[databricks]
cryptography==1.5

Status on running clusters

[✓] Install automatically on all clusters

Status	Cluster Name	Message
● Installed	jp-databricks-cluster-4…	

Figure 7: PyPI package on databricks cluster. Left: User's guide, Right: configuration used for this case study.

RESULTS

The result of this study is an enhanced demo version compared to the one reported in case study 212. If you have an Azure subscription, you can build the demo by yourself starting here. If you need a step-by-step tutorial on how to deploy Azure ML SDK and databricks, you can use this youtube video.

PERFORMANCE BENCHMARKING

This demo was run on a databricks cluster with 2 to 8 nodes that are Standard_DS13_v2: the table below provides performance figures for the current and previous release, which was tested on a single DSVM with 4 vCPUs and 16 GB RAM. *Use these figures to get a feel for improvement and not as absolute values.*

Release	Notebook 2 Time-to-solution	Notebook 3 Time-to-solution
previous	71 minutes	10 minutes
current	27 minutes	21 minutes

CONCLUSION & RECOMMENDATIONS

As recommended by Microsoft, we have run the predictive maintenance application using the new software architecture including Azure Machine Learning SDK and Azure Databricks.

The data, features set, labels and model are the same as in the previous release, therefore the performance is also approximately the same.

The significant advantages of the current release compared to the previous one, are:

- *Shorter time to solution*, reported in the paragraph Performance Bench-marking.
- *Easier administration,* because the entire application runs in Azure, i.e. the client-side workbench component has been removed.

Predictive Maintenance is one of the most popular ML applications, and one that is widely considered prime time for productive deployment: custom solutions exist, as well as off-the-shelf packages, and software building blocks that may be sold with consulting services. We have recently met with several companies across the world that either implement or propose predictive maintenance for production. These companies are SEW, Duerr, Compacer, Siemens, SAS, and more.

One benefit of this case study is that it is built on open source software, and hence it can be adapted to support a customer's proof of concept for predictive maintenance of machines, industrial or end-user equipment, plants, etc.

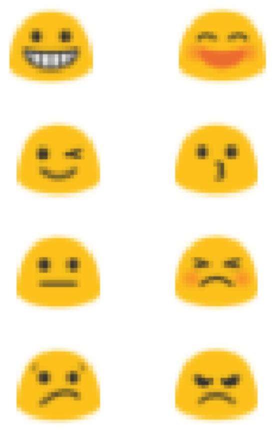
While the available results are promising in a demo environment, a significant effort to customize the model for real use cases would be needed, especially to build features that effectively classify failures. This effort requires both data science skills and domain expertise.

Potentially, Microsoft recently announced Automated Machine Learning could help to optimize the model precision by iteratively determining the best hyperparameters in a very effective, automated way in order to accelerate time-to-market for the application.

If you are interested to implement predictive maintenance in your environment using ML, we can offer a discovery workshop and Scope of Work service in collaboration with industry partners Microsoft and Nvidia if appropriate. Please approach joepareti54@gmail.com.

Team 221

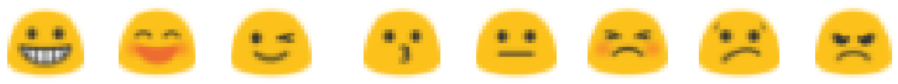
Consumer Analytics using Natural Language Processing and Artificial Intelligence in the Cloud



"UberCloud's HPC and Cloud computing capabilities have aided in the processing of large volumes of consumer data in order to build AI models for making better decisions and game-changing strategies."

MEET THE TEAM
End-User/Data Science Expert: Veena Mokal, Data Science Expert, MBA in Business Analytics, Institute of Management Technology, INDIA
Software Provider: Anaconda Python distribution platform
Resource Provider: On-premises systems. And next step: UberCloud Engineering Simulation Platform
HPC Expert: Praveen Bhat, HPC/Python Techn Consultant, India, Wolfgang Gentzsch, UberCloud
This project has been performed in 2021.

USE CASE
In recent years, advancements in internet connectivity have brought significant opportunities to customers and shoppers. Because of these advancements in internet connectivity, rapidly rising e-commerce enterprises have yielded true big data. The enormous popularity of big data on social media allows purchasers to voice their opinions and views on a wide range of topics such as the status of the economy, or to express their **displeasure** with certain items or services, or to express their **satisfaction** with their purchases.



Such numerous consumer opinions and product reviews contain rich and valuable information and recently became important sources for both consumers and business firms. Consumers commonly seek quality information from online reviews before purchasing a product, while many firms use online reviews as important feedback of their products, marketing and consumer relationship management. Therefore, understanding the psychology behind online consumer behaviour became the key to compete in today's markets which are characterized by ever-increasing competition and globalization.

Sentiment analysis & text analysis are the applications of big data analysis, which aim to aggregate and extract emotions and feelings from different kinds of reviews. These big data which is growing exponentially are mainly available in an unstructured format, and they are not machine-processable and interpretable. Therefore, the use of the Natural Language Processing (NLP) machine learning technique is essential which focuses on extracting this data and opinions from the huge amount of information present on the web.

Consumer sentiment analysis is one of the popular techniques of discovering the emotion to understand your customer related to your product or service. The scope of this project is to develop a consumer analytics framework considering e-commerce review data using AI machine learning techniques with a Cloud solution capability. This study enables the objective to understand the consumer in real-time and to identify a critical issue affecting the business. As this process is resource-driven, and the data extracted from social media is huge, it requires a higher number of CPU cores and RAM for faster data processing, model building & data visualization. This study explores this relationship through a cloud-based solution infrastructure to speed up computation time and achieve the faster turn-around time required in the model building activity.

PROCESS OVERVIEW

The following defines the step-by-step approach in setting up the model environment for consumer analytics using NLP and Python.

a. Data Pre-Processing & Feature Extraction

The pre-processing of data involves Text cleaning & Feature Extraction:

Text Cleaning: Text cleaning activity is one of the major steps in data pre-processing. In this step punctuations, stop words, URL Links, HTML tags, and special characters like emoticons and emoji are removed and the text converts into lower case. In the final step, text cleaning performs spell checks and corrects it grammatically.

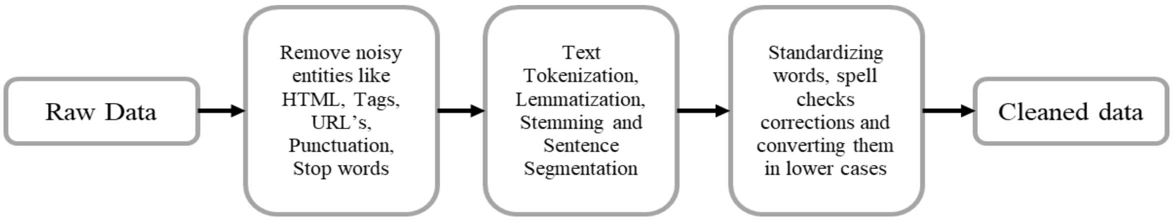

Raw Data
Remove noisy entities like HTML, Tags, URL's, Punctuation, Stop words
Text Tokenization, Lemmatization, Stemming and Sentence Segmentation
Standardizing words, spell checks corrections and converting them in lower cases
Cleaned data

Figure 2: Text cleaning process

Feature Extraction: The cleaned data is then processed to extract features (variables) that help to understand the distribution of the review text. This includes the distribution of characters, numerical values, lowercase words, and average word length. This completes the data pre-processing step.

b. Performing Sentiment Analysis

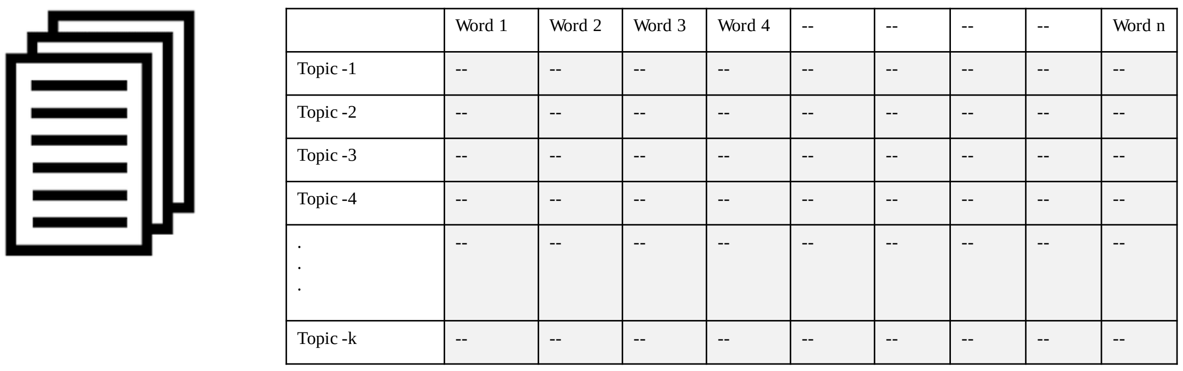
This section mainly focuses on generating sentiment scores for the review text data. Sentiment score is a function of polarity & subjectivity. Both parameters are extracted from the review text using NLP algorithms to understand the overall sentiment. Typically, the overall sentiment is often inferred as positive, neutral or negative from the sign of the polarity score. Polarity is a floating-point number that lies in the range of [-1,1] where 1 means a positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float that lies in the range of [0,1].

c. Topic Modeling

Topic modelling is the process of identifying topics in a set of documents. It enables search engines on the topics of documents that are important. There are multiple methods of doing this, however, this project includes Latent Dirichlet Allocation (LDA). LDA is a form of unsupervised learning that views documents as bags of words. It works by first making a key assumption: the way a document was generated was by picking a set of topics and then for each topic picking a set of words. To do this it does the following for each document m:

The following algorithm steps are implemented using Python and LDA for topic modelling:

- Assume there are k topics across the whole document
- Distribute these k topics across document m (this distribution is known as α and can be symmetric or asymmetric, more on this later) by assigning each word a topic
- For each word w in document m, assume its topic is wrong, but every other word is assigned the correct topic
- Probabilistically, assign word w a topic based on two things:
    - what topics are in document m
    - how many times word w has been assigned a particular topic across all the documents?
- Repeat this process several times for each document to get the list of topics



	Word 1	Word 2	Word 3	Word 4	--	--	--	--	Word n
Topic -1	--	--	--	--	--	--	--	--	--
Topic -2	--	--	--	--	--	--	--	--	--
Topic -3	--	--	--	--	--	--	--	--	--
Topic -4	--	--	--	--	--	--	--	--	--
. . .	--	--	--	--	--	--	--	--	--
Topic -k	--	--	--	--	--	--	--	--	--

Figure 3: Details on topic modelling structure

The above figure shows the representation of topic modelling structure and how the individual topics are related to each of the words in the documents.

d. Predictive modelling

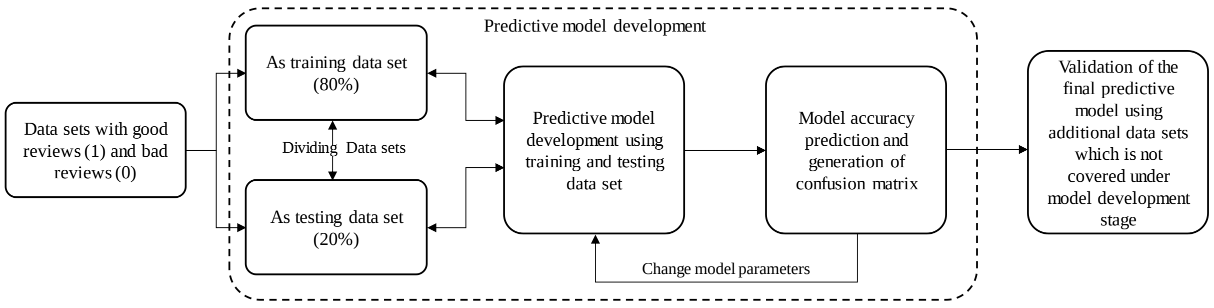

Data sets with good reviews (1) and bad reviews (0)
Predictive model development
As training data set (80%)
Dividing Data sets
As testing data set (20%)
Predictive model development using training and testing data set
Model accuracy prediction and generation of confusion matrix
Change model parameters
Validation of the final predictive model using additional data sets which is not covered under model development stage

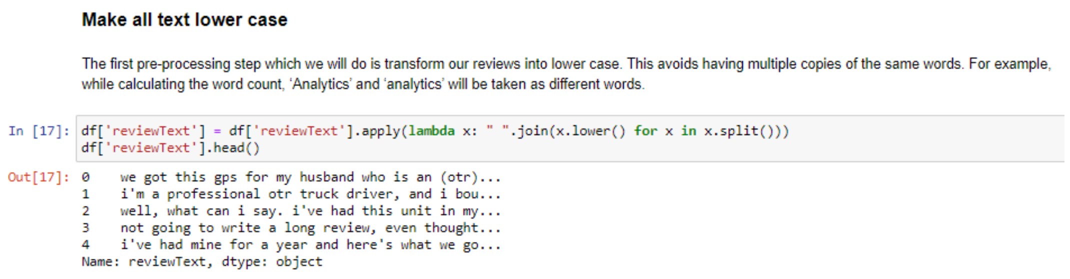
Figure 4: Predictive model development framework

The objective of this phase is to develop a modelling methodology that classifies new input review text into good or bad reviews. The classification accuracy and validation of the model become key criteria for the selection. The predictive model can be developed using both supervised and unsupervised learning methods. This study covers the following predictive modelling techniques to predict the type of reviews (good or bad reviews):

- Naïve Bayes with
  - Gaussian method
  - Multinomial method
  - Bernoulli method
- Logistic Regression model fitting

RESULTS & DISCUSSION

This section details out the results extracted by running the Python scripts developed for the methodology explained. The following provides details on the results derived and insights gained from the analysis.



Make all text lower case

The first pre-processing step which we will do is transform our reviews into lower case. This avoids having multiple copies of the same words. For example, while calculating the word count, 'Analytics' and 'analytics' will be taken as different words.

```
In [17]: df['reviewText'] = df['reviewText'].apply(lambda x: " ".join(x.lower() for x in x.split()))
         df['reviewText'].head()

Out[17]: 0    we got this gps for my husband who is an (otr)...
         1    i'm a professional otr truck driver, and i bou...
         2    well, what can i say. i've had this unit in my...
         3    not going to write a long review, even thought...
         4    i've had mine for a year and here's what we go...
         Name: reviewText, dtype: object
```

Figure 5: Example showing texts converted to lower case

The first step shows all the review texts converted into lower case followed by removing the punctuation marks and stop word removal, unwanted texts like HTML tags, emoticons, white spaces etc. The data set is further subjected to multiple pre-processing steps involved which will further help to cleanse and structure the data sets for further analysis. This includes - extracting the number of words, characters and average word length.

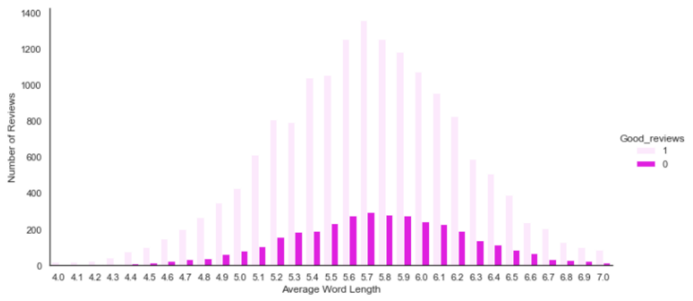


Figure 6: Distribution showing the average word length for good and bad reviews

The above figure shows the average word length for good and bad reviews. From the distribution, the word length for a good review is longer than the bad review and hence the processing time for the good review data is higher when compared to the bad review data.

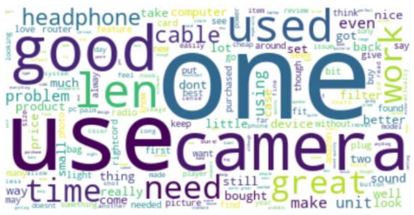


Figure 7: Word Cloud extracted from the processed review data

Figure 7 shows the word cloud which is a visual representation of the word frequency. The more commonly the term appears within the text being analysed, the larger the word appears in the image generated. Word clouds are increasingly being employed as a simple tool to identify the focus of written material.

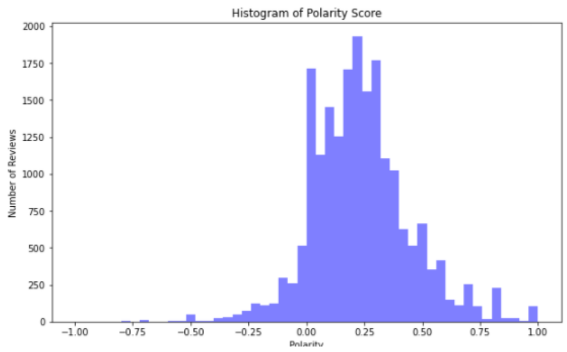


Figure 8: Sentiment scores for the first 2000 reviews

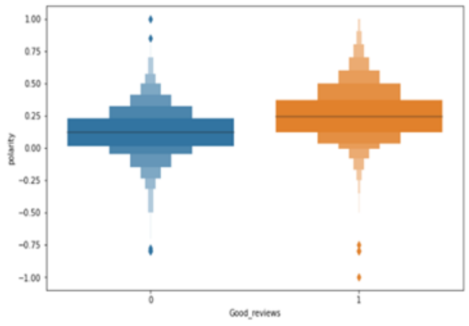


Figure 9: Polarity values for bad (0) & good reviews (1)

Figure 8 shows the sentiment score distribution for the first 2000 reviews where the major portion of the review sentiments shows positive polarity for good reviews. *Figure 9* shows a good review distribution on polarity. Good reviews that have low polarity are categorized in negative sentiments. Bad reviews which have high polarity are categorized in positive sentiments.

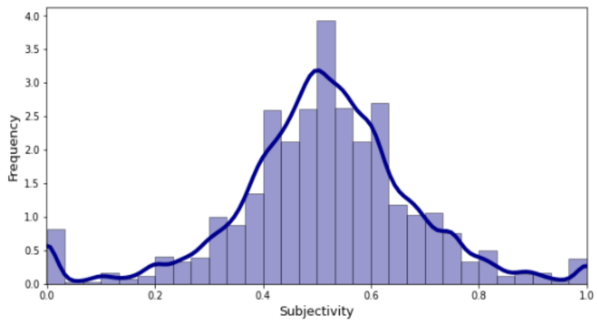


Figure 10: Distribution of subjectivity

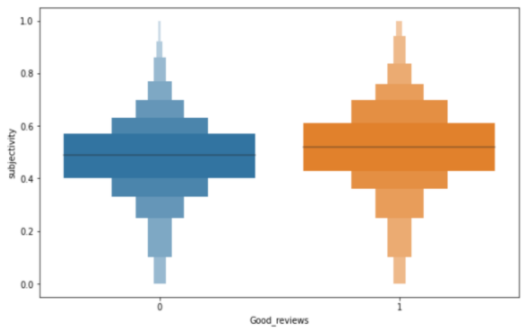


Figure 11: Subjectivity values for good (1) and bad (0) reviews

Figure 10 shows the subjectivity score distribution. *Figure 11* shows a good review distribution on subjectivity. *Figure 12* shows the distribution of subjectivity and polarity for good and bad reviews.

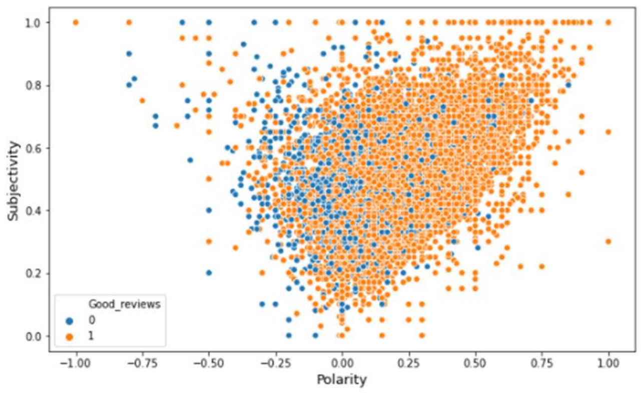
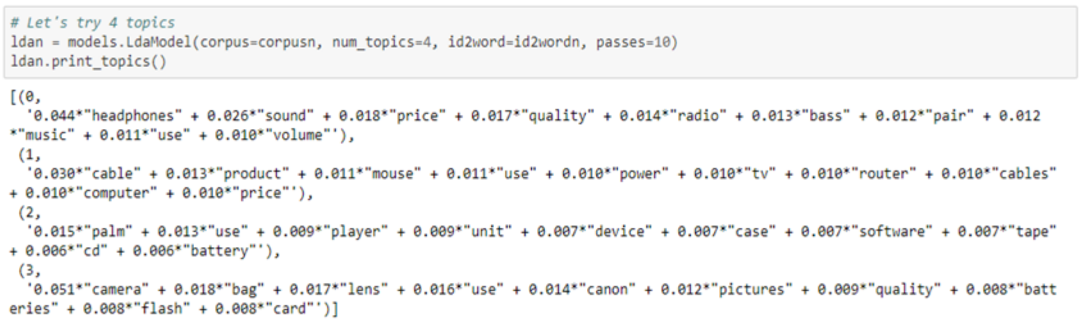


Figure 12: Distribution of subjectivity and polarity for bad and good reviews

The topic modelling activities focus on defining the list of topics from the review data and creating a matrix of topics. The LDA algorithm is used to analyse the topics and generate the probability of occurrence of topics in a document based on the words. The LDA works with an assumption that each document is a collection of words, “bag of words”, thus the order of the words and their grammatical role are not considered in the model, providing the LDA equation considering 4 topics.



```
# Let's try 4 topics
ldan = models.LdaModel(corpus=corpusn, num_topics=4, id2word=id2wordn, passes=10)
ldan.print_topics()
```

```
[(0,
  '0.044*"headphones" + 0.026*"sound" + 0.018*"price" + 0.017*"quality" + 0.014*"radio" + 0.013*"bass" + 0.012*"pair" + 0.012
*"music" + 0.011*"use" + 0.010*"volume"'),
 (1,
  '0.030*"cable" + 0.013*"product" + 0.011*"mouse" + 0.011*"use" + 0.010*"power" + 0.010*"tv" + 0.010*"router" + 0.010*"cables"
+ 0.010*"computer" + 0.010*"price"'),
 (2,
  '0.015*"palm" + 0.013*"use" + 0.009*"player" + 0.009*"unit" + 0.007*"device" + 0.007*"case" + 0.007*"software" + 0.007*"tape"
+ 0.006*"cd" + 0.006*"battery"'),
 (3,
  '0.051*"camera" + 0.018*"bag" + 0.017*"lens" + 0.016*"use" + 0.014*"canon" + 0.012*"pictures" + 0.009*"quality" + 0.008*"batt
eries" + 0.008*"flash" + 0.008*"card"')]
```

Figure 12: LDA models with 4 topics considered

A similar approach was followed to extract the topics using both nouns and adjectives and then use the LDA algorithm to analyse the topics. The above exercise is performed for both good and bad reviews separately to generate a list of topics and the associated bag of words for further analysis.

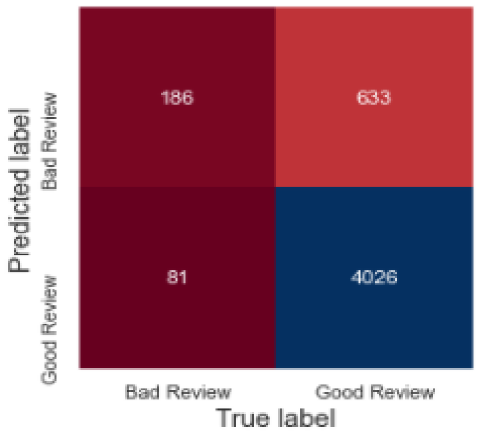


Figure 13: Confusion matrix for logistic regression

The processed data which contains good and bad reviews groups are used for training the predictive models. The review dataset is divided into an 80% training and a 20% testing data group which is used for building the model. The following modelling output is evaluated for model accuracy through a confusion matrix and an accuracy percentage. The Gaussian predictive model shows 71% of prediction accuracy, where around 271 data points are misclassified for the good review and 1142 data points are misclassified for the bad review.

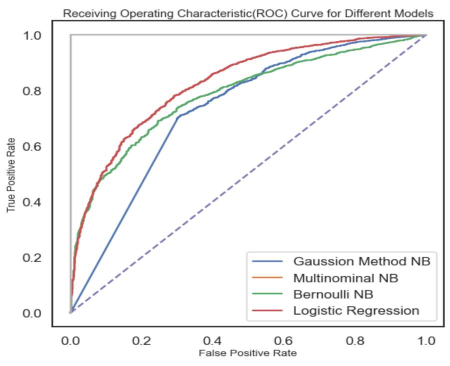
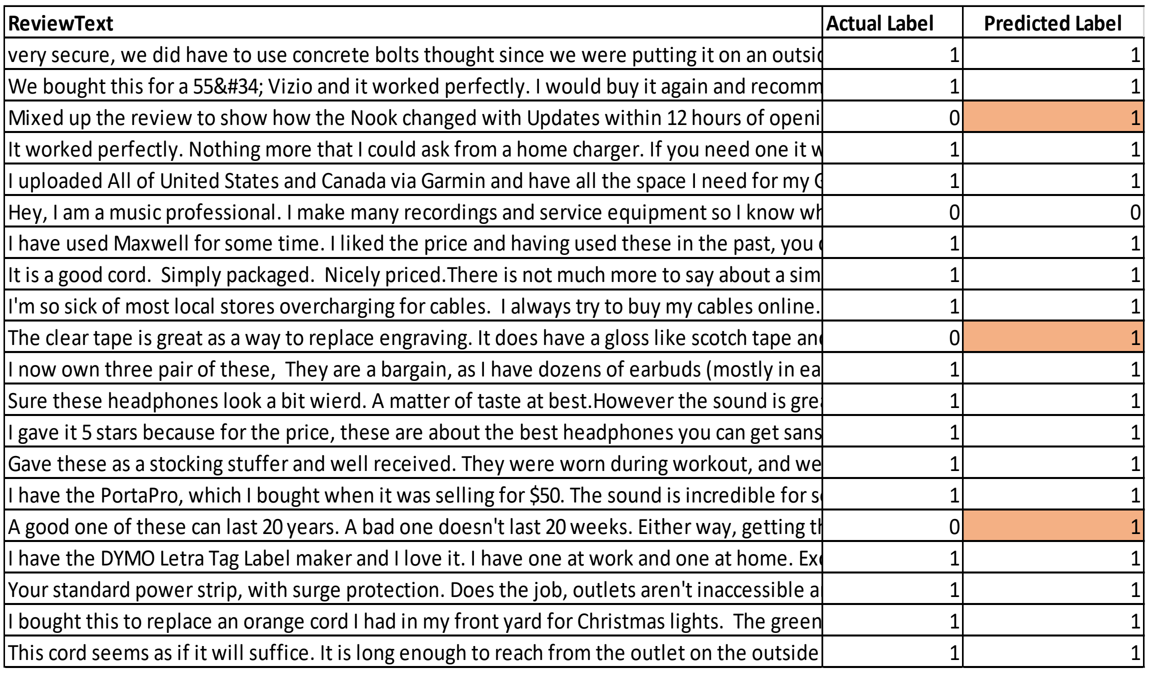


Figure 14: Comparison of different modelling techniques

The Multinomial predictive model shows 84% of prediction accuracy where around 749 data points are misclassified as the good review and 21 data points are misclassified as the bad review. The Bernoulli predictive model shows 80% of prediction accuracy where 450 data points are misclassified as the good review and 559 data points are misclassified as the bad review. The Logistic regression predictive model shows 86% of prediction accuracy where 633 data points are misclassified as the good review and 81 data points are misclassified as bad review. Out of all predictive models, the logistic regression model has a better prediction in terms of accuracy with a lesser misclassification percentage. *Figure 14* shows the comparison done on different predictive modelling methodology.

MODEL VALIDATION

The logistic regression model is validated using the latest review data from the e-com website. The data is then pre-processed using the Python script and analysed using the logistic regression model.



ReviewText	Actual Label	Predicted Label
very secure, we did have to use concrete bolts thought since we were putting it on an outsi	1	1
We bought this for a 55&#34; Vizio and it worked perfectly. I would buy it again and recomm	1	1
Mixed up the review to show how the Nook changed with Updates within 12 hours of openi	0	1
It worked perfectly. Nothing more that I could ask from a home charger. If you need one it w	1	1
I uploaded All of United States and Canada via Garmin and have all the space I need for my G	1	1
Hey, I am a music professional. I make many recordings and service equipment so I know wh	0	0
I have used Maxwell for some time. I liked the price and having used these in the past, you	1	1
It is a good cord. Simply packaged. Nicely priced.There is not much more to say about a sim	1	1
I'm so sick of most local stores overcharging for cables. I always try to buy my cables online.	1	1
The clear tape is great as a way to replace engraving. It does have a gloss like scotch tape an	0	1
I now own three pair of these, They are a bargain, as I have dozens of earbuds (mostly in ea	1	1
Sure these headphones look a bit wierd. A matter of taste at best.However the sound is gre	1	1
I gave it 5 stars because for the price, these are about the best headphones you can get sans	1	1
Gave these as a stocking stuffer and well received. They were worn during workout, and we	1	1
I have the PortaPro, which I bought when it was selling for $50. The sound is incredible for s	1	1
A good one of these can last 20 years. A bad one doesn't last 20 weeks. Either way, getting t	0	1
I have the DYMO Letra Tag Label maker and I love it. I have one at work and one at home. Ex	1	1
Your standard power strip, with surge protection. Does the job, outlets aren't inaccessible a	1	1
I bought this to replace an orange cord I had in my front yard for Christmas lights. The green	1	1
This cord seems as if it will suffice. It is long enough to reach from the outlet on the outside	1	1

Figure 15: Latest data extracted (Jan-Feb 2021) from e-com portal where actual and predicted labels are compared

Prediction accuracy for good review data has 79 misclassified data out of 4000 good review data points.

HPC PERFORMANCE BENCHMARKING

The NLP – Machine Learning algorithm for e-com reviews is a very compute intensive technique, therefore, to complete the study, we have run a performance analysis using a high-performance desktop machine that has 16 CPU Cores and 32 GB RAM. The performance analysis was conducted to study the computing system requirement to run millions of review data.

CHALLENGES

The challenges faced in the project were related to processing massive volumes of data from different social media and e-commerce websites. The project requirement was to build an AI-driven model to process the reviews from different web sources and analyse the sentiments behind these reviews. Processing review data with local computers was a challenge to handle the data size. This resulted in the need to use a high-performance compute node, where we could load the big volume of data to perform data analysis and develop required steps to pre-process the data.

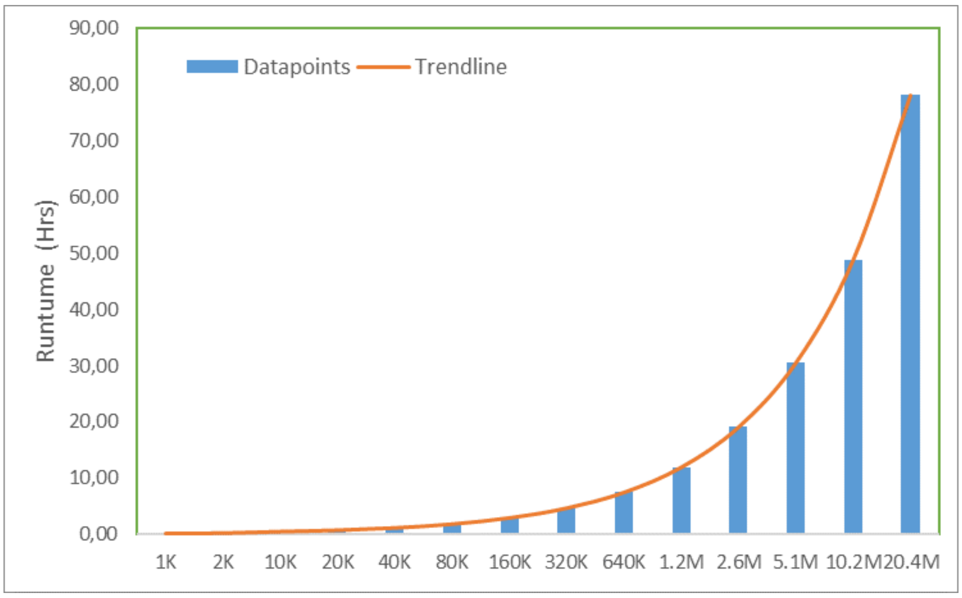


Figure 16: Compute runtime captured for different sizes of review data sets

BENEFITS

1. The HPC cloud computing environment features the Python-based Anaconda platform that aided in data analysis and the construction of predictive models. Dealing with a large volume of data and pre-processing formatting activities was difficult in this project. These tasks demanded a significant amount of computing power. The handling and processing of such massive amounts of data was made possible by cloud HPC.
2. Experiments conducted in the HPC Cloud environment demonstrated the ability to remotely set up and run Big Data analysis as well as build AI models in the cloud. The AI - Machine learning model setup requirements were pre-installed in the HPC container, allowing the user to access the tools without installing any kind of prior set up.

CONCLUSION & RECOMMENDATIONS

- Advanced machine learning technology, such as NLP, is a field of study that examines people's sentiments, attitudes, or emotions toward specific entities. This study addresses the fundamental problem of consumer behavior by using sentiment analysis, sentiment polarity categorization, and high-performance computing containers to speed up the process.

- All of these applications show how sentiment analysis can be a useful resource for analyzing affective information in social platforms, relying not only on domain-specific keywords but also on commonsense knowledge bases that allow for the extrapolation of cognitive and affective information associated with natural language text.
- UberCloud's HPC resource was a good fit for performing NLP analytics and building AL-ML models involving social media and e-commerce big data that could not be executed on a standard workstation. UberCloud's environment aided in speeding up model development activities within a set timeline and completing the project successfully.

Cloud to Cloud. The Cloud operators or IT departments no longer need to limit the variety, since they no longer have to install, tune and maintain the underlying software. They can rely on the UberCloud Containers to cut through this complexity. This technology also provides hardware abstraction, where the container is not tightly coupled with the server (the container and the software inside isn’t installed on the server in the traditional sense). Abstraction between the hardware and software stacks provides the ease of use and agility that bare-metal environments lack. Finally, these containers run on top of the UberCloud Engineering Simulation Platform, which is fully automated, self-service, and are easily migrated from any cloud to any other cloud.

Team 222

A Personalized Approach Towards Repairing Cardiac Valve Leakage
Patient-Specific Data Generation and Machine Learning Computations

Yaghoub Dabiri, Scientist at 3DT Holdings, San Diego
Wolfgang Gentzsch, Co-founder and President, UberCloud
Julius Guccione, Professor of Surgery at the UCSF Medical Center, San Francisco
Ghassan Kassab, President at California Medical Innovations Institute, San Diego

Significance

Mitral regurgitation (MR) is the most common cardiac valvular disease in the U.S. Approximately 4M people have severe MR, and roughly 250,000 new diagnoses of MR are made each year especially in the older population. Nearly half of patients identified with moderate-severe MR cannot undergo surgical repair of the mitral valve (MV) due to frailty, other co-morbidities, and/or impaired left ventricular (LV) function. In 1998, Fred St Goar and colleagues developed a percutaneous, catheter-delivered implantable clip, applied to the MV for the treatment of both functional and degenerative MR (now market-approved in Europe/USA). A five-year clinical trial study on MitraClip (MC) alone and MC plus annuloplasty ring, known as EVEREST (Endovascular Valve Edge-to-Edge Repair Study) suggested various short-term and long-term benefits of MC therapy.

MitraClip Procedure

Mitral regurgitation (MR) condition cannot be medically treated, and previously could only be repaired with open-heart surgery on patients who were otherwise physically healthy. MitraClip brings a minimally invasive alternative to open-heart surgery. The MitraClip is a small metal clip that helps patients with mitral regurgitation (MR), a condition where the heart’s mitral valve leaflets do not close tightly, causing blood to leak into the heart’s left atrium and can lead to advanced heart failure. This new treatment expands the options for selected patients with MR, especially those who are not candidates for invasive open-heart surgery. The procedure allows doctors to use catheter-based technology to repair the mitral valve, without the need for patients to undergo cardiopulmonary bypass. The MitraClip procedure shortens recovery time and ultimately improves quality of life for those experiencing life-altering symptoms like fatigue and shortness of breath. During the MitraClip procedure, a physician will use traditional catheter methods to guide the clip into the left atrium. The clip is lowered and attached to the valve to repair or reduce MR. Before final placement, the clip can be moved and rotated to ensure optimal fit.

However, there are several potential issues with the current ad hoc method of MC placement, including: (1) No efficacy (i.e., no reduction of insufficiency, e.g., improper positioning of MC placement or single leaflet attachment); (2) Damage of mitral leaflets or MC entangled in chords making repair at a later date very difficult; (3) Perforation of the MV leaflet; and (4) Creation of mitral stenosis. Considering the MC cost of $30,000 and 250,000 annual new cases, any improvements in MC placement may translate to potential savings of $2.8B.

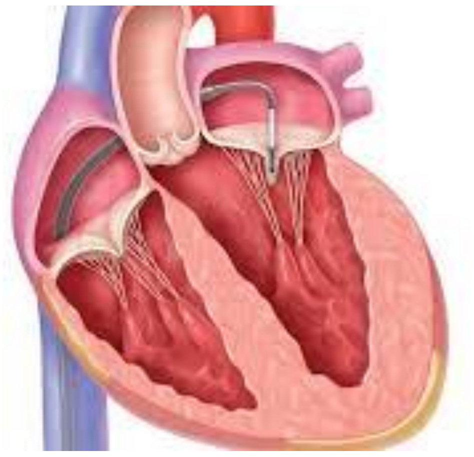


Fig. 1: MitraClip Procedure for Mitral Regurgitation

Our software tool will be a personalized approach towards improving the outcomes of MC intervention. This tool will accomplish the following: (1) Determine the optimal number of MCs and their positions that minimize MR; (2) Assess the likelihood of leaflet perforation and tear; and (3) Evaluate the efficacy of the MC at the delivered position if the optimal MC position is missed. Such a tool will enable interventional cardiologists to make an intelligent decision on the level of the MC benefit in real time before retracting the delivery catheter.

An In-silico Personalized Approach

Using patient-specific images, we create physics-based models for each individual. These in-silico models are created using the Abaqus finite element package. As shown in Fig. 2, the regurgitant flow through the MV (valve leakage) can be determined for different MC scenarios including number of clips and their locations. This approach is in line with the FDA vision to establish in-silico clinical trials for medical device analysis.

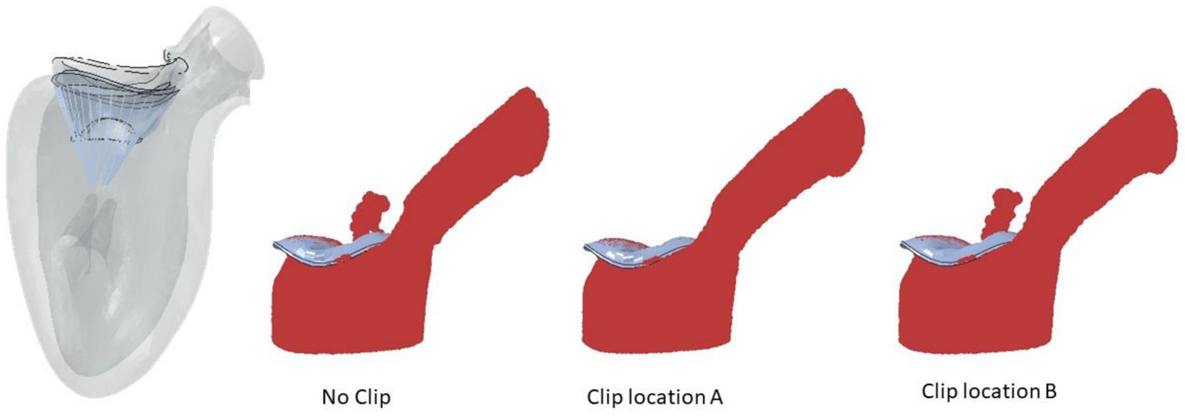


Fig. 2: The in-silico models provide the MR for different clip scenarios. Here the valve leakages (blood is shown in red) for two possible clip locations are shown.

Machine Learning Analysis

Our team has extensive experience in using machine learning in cardiac applications. In two recent peer-reviewed papers [8,9] we used machine learning to predict important parameters such as left ventricle pressure, volume and stresses, in a matter of seconds. Using physics-based computational modeling, these results require much longer time (on average 12 – 24 hours per simulation). For example, one full cardiac cycle requires nearly two hours to provide pressure and flow in the left ventricle, but we estimated the same data using machine learning in less than 5 seconds. The accuracy of our machine learning models were over 95% compared to physics-based ground truth data.

Our recently-published paper describes the details of our approach for predicting cardiac mechanics using feed forward and recurrent deep learning algorithms. We used both TensorFlow and Pytorch in our computations. We were able to produce results with higher accuracy using deep learning compared to XGBoost (eXtreme Gradient Boosting library) (Fig. 3). Based on preliminary results, we plan to predict the outcomes of MC intervention with a similar accuracy (higher than 90%), providing the interventional cardiologists with high confidence about the optimal MC implantation strategies.

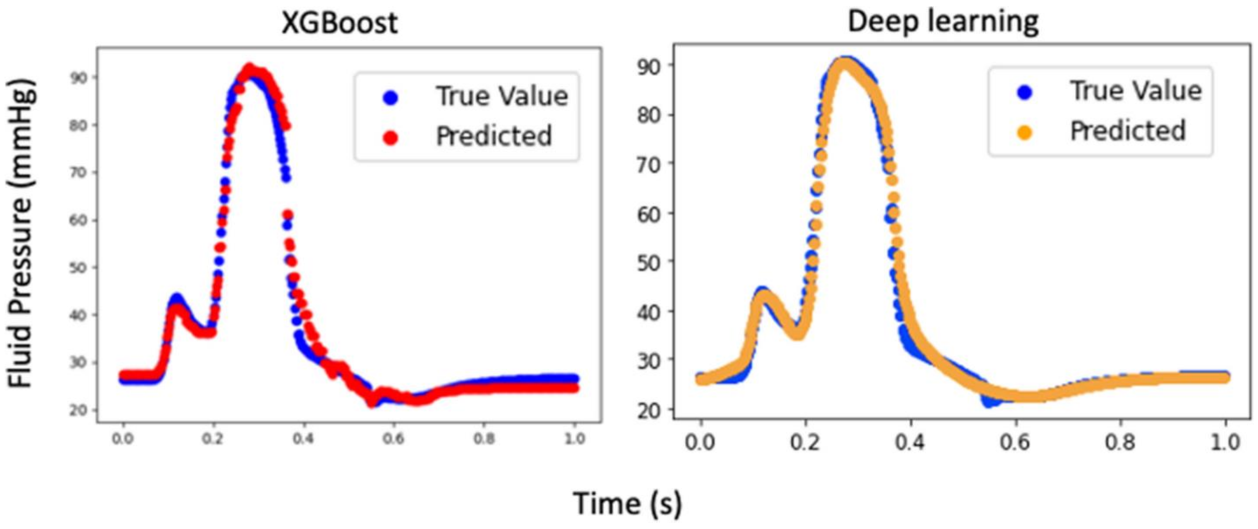


Fig. 3: A comparison between results using XGBoost and deep learning, for prediction of blood pressure in the left ventricle during a cardiac cycle [9]. Deep learning could provide results with higher accuracy. The deep learning error was 0.687 ± 0.165 ml whereas the XGBoost error was 1.734 ± 0.584 [9].

Smoothed Particle Hemodynamics

We use a capability implemented in Abaqus explicit for modeling the blood flow, known as smoothed particle hydrodynamics (SPH). In traditional finite element methods, the domain is divided into meshes. This approach is not currently efficient to model the interaction between blood flow and valve tissue due to complex geometry, contact and boundary conditions, leading to highly distorted problematic elements. In SPH the blood domain is represented by particles. These particles can handle loads and boundary conditions needed to create the MC intervention model.

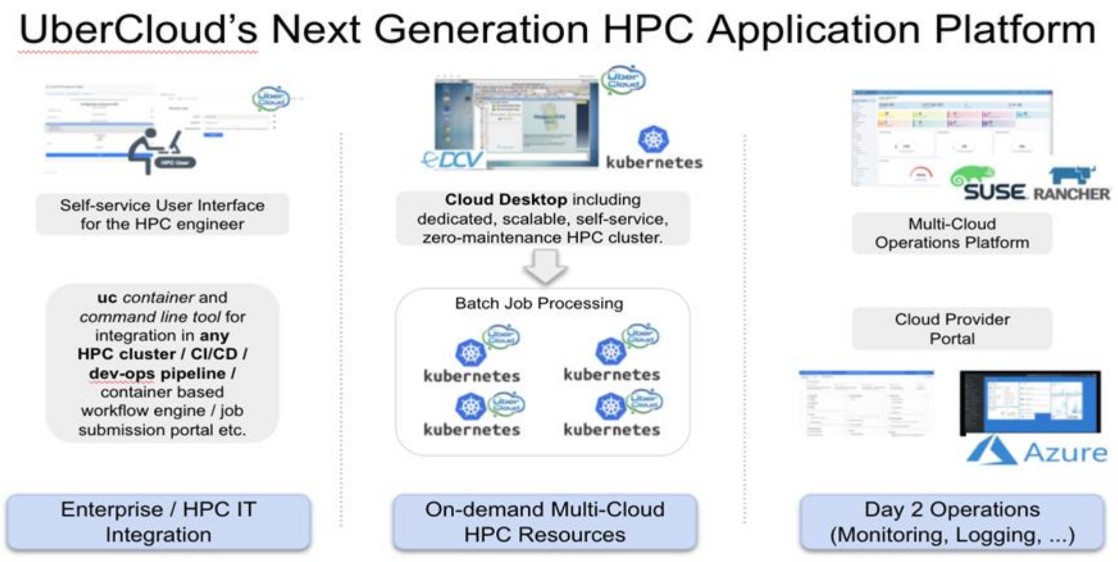
Modeling Workflow

We created a dataset from finite element models. Each finite element model represents the MV regurgitation for a distinct MC implementation strategy and MV geometry. From each simulation, we postprocess the regurgitation flow and other parameters such as leaflet stresses. After all simulations are performed, we will have the MV data for different MV geometries and MC implantation strategies. Using this dataset, ML models namely neural networks (deep learning) and decision tree (XGBoost) algorithms will be used to predict outcomes of different MC strategies. The dataset will be divided into train and test sets. Using test data (which are not seen during training), the accuracy of ML models will be assessed based on criteria such as mean absolute error and coefficient of determination. As mentioned above, the ML is expected to provide results in a matter of seconds which is much shorter than finite element runtime (several hours).

Technical aspects of the model's runtime

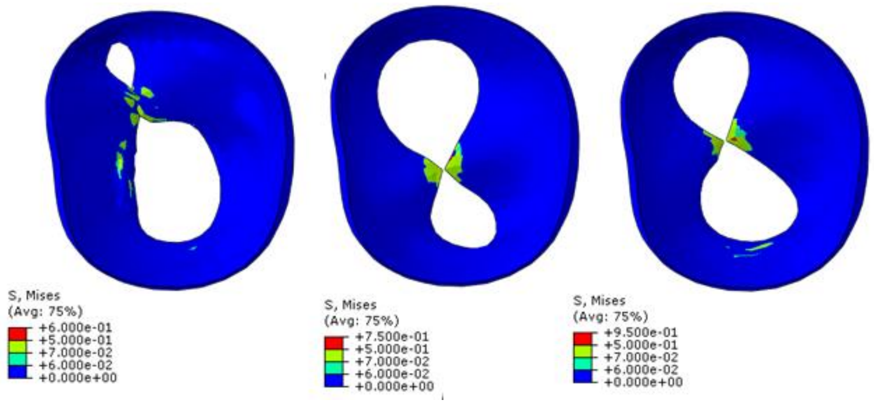
We use Abaqus 2020 explicit for finite element calculations. The runtime for each model depends on the geometry of that model. The input file is around 23 MB (for each job there are a couple of input files), and the generated odb files are around 2.3 GB. We will run batches of jobs (e.g., 50 at a time) to monitor how well they run.

Recently UberCloud has been challenged by 3DT Holdings, a spinoff of the University of California San Diego, with the task of simulating cardiac valve leakage with patient-specific data generation and Machine Learning on an Azure,/Google Cloud Platform multi-cloud environment, with SUSE Rancher Kubernetes management and with UberCloud engineering simulation platform.


UberCloud's Next Generation HPC Application Platform
Self-service User Interface for the HPC engineer
uc container and command line tool for integration in any HPC cluster / CI/CD / dev-ops pipeline / container based workflow engine / job submission portal etc.
Enterprise / HPC IT Integration
eDCV
kubernetes
Cloud Desktop including dedicated, scalable, self-service, zero-maintenance HPC cluster.
Batch Job Processing
kubernetes
kubernetes
kubernetes
kubernetes
On-demand Multi-Cloud HPC Resources
SUSE RANCHER
Multi-Cloud Operations Platform
Cloud Provider Portal
Azure
Day 2 Operations (Monitoring, Logging, ...)

At UberCloud we supported the end user Yaghoub Dabiri from 3DT Holdings, Julius Guccione, UCSF, and Ghassan Kassab, UCSD, with running 3,000 complex living heart simulations, accurately simulating blood flow, stress, volume, and pressure inside a human heart. The goal of this project was to find the optimal position of a so-called MitraClip that's placed in the heart's mitral valve to reduce mitral regurgitation (MR). A MitraClip is a small metal clip that allows doctors to perform catheter-based surgery to repair the mitral valve, and that brings a minimally invasive alternative to open-heart surgery. With these 3,000 simulations we collected enough simulation results in order to train a set of ML algorithms which can then predict the best suited MitraClip position at the valve in almost real time (within a couple of seconds) instead of hours or even days.

In order to stay within 3DT Holdings' budget we used Google Cloud Platform's C2 HPC VMs as they can be allocated in preemptible mode. But since we had to manage and monitor the environment on behalf of our customer, some of the required architecture components had been implemented within our own Azure account. While saving 80% of the costs, preemptible instances have no guaranteed uptime and can only stay running for a maximum of 24h. Hence our decision was to run each simulation in its own Google Kubernetes Engine cluster which is automatically started each time from scratch to reduce the risk of letting a simulation get interrupted. In this specific case the cluster startup time didn't have much effect as each simulation has to be run for several hours. The remaining infrastructure, like license servers and SUSE's Rancher for monitoring purposes has been running on Azure. Both environments have been protected by firewall rules, only allowing traffic flowing between the components. One special GKE cluster was not using preemptible but cheaper instance types and a GPU. This cluster has been used to run UberCloud's Abaqus container, providing a remote (accessible via NI-SP's DCV) Linux desktop with Abaqus installed. This cluster served the whole project as an entry point for the research engineer, uploading and preparing the input data and supervising the output results.



Stresses on the mitral valve for different MitraClip implant placements

While managed Kubernetes deployments added some slight overhead (management fee, more daemons / processes running on compute nodes, ...), the advantages of having 80% cost reduction on C2 preemptible CPU price, using a consistent application stack deployed by a single UberCloud command, and exploiting compatibility with the whole Kubernetes ecosystem by our SUSE Rancher integration outweighed them by far. We minimized costs, and job failures due to preemption, while at the same time we could maximize cloud resource and license usage to get maximum throughput. That way, the total cost of the 3,000 simulations stayed under $20K.

This project is a collaboration between UberCloud, 3DT Holdings LLC, University of California San Francisco, National Heart Center Singapore, Dassault Systèmes Simulia, the Teams at Google Cloud Platform, Microsoft Azure, and SUSE Ranger.

Please contact Simr at contact@simr.com before distributing this material in part or in full.
© Copyright 2024 Simr™. Simr is a trademark of TheUberCloud Inc.