Anomaly Detection and Predictive Maintenance

Wolfgang M. Pauli, Ph.D.

> wolfgang.pauli@microsoft.com

Seth Mottaghinejad

> sethmott@microsoft.com

Getting started

- introduction of the Learn-AI team
- instructor introductions and backgrounds
- quick survey of participant backgrounds and expectations from the course

Access to the course environment

- we recommend that you open your browser to a **private (incognito) mode** to follow these instructions
- please log into https://aka.ms/learnai-adpm and fill out registeration form and hit `Submit` followed by `LAUNCH LAB`
- after a few minutes, you will get a set of access credentials
  - with the **Microsoft Azure** user credentials log into the Azure portal and examine the resources that were created
  - using **Environment Details** credentials, go to http://<DSVM_NAME>:8000 and log in with the user name and password provided
- save your credentials somewhere in case you accidentally close the page and need to log back in

Course series: Using AI to detect and predict anomalies in structured and unstructured time series data

- Course 1: Anomaly detection on structured streaming data (*today*)
- Course 2: Predictive maintenance using automated machine learning (*tmrw*)
- Course 3: Anomaly detection on unstructured (video) data (*in progress*)
- Course 4: Deploying Anomaly detection to IoT/Edge devices (*in progress*)

About the course

- you are presented with many data science-*ish* challenges throughout the course
- you learn about different tools in the process of building an E2E solution
- this course is hands-on with lots of exercises (solutions will be provided)
- during exercises you're encouraged to **sit together and work in groups**
- a few of the exercises are intentionally longer and difficult
- programming exercises are in Python on a Jupyter Notebook, so intermediate knowledge of Python is required

Learning goals

- understand how to combine Azure AI Platform tools to solve real-world problems
- learn to think like a data scientist and speak the language
- learn to follow best practices to handle complex projects
- use automated machine learning to improve on hand-trained models
- use the Azure SDK to manage Workspaces and train and deploy models

Tools covered

- Azure Machine Learning (AML) Python SDK
  - HyperDrive (for hyper-parameters tuning)
  - AmlCompute (fka BatchAI)
  - automated Machine Learning
- Two flavours of Data Science notebooks
  - Azure Databricks
  - Jupyter Notebooks
- Python (esp. `pandas` and `scikit-learn`) and Conda
- Open-Source Python port of Twitter anomaly detection API

Jupyter Notebooks

- A document with cells containing Markdown, Python, bash and more
- A code cell shows `In []` when not yet run and `In [*]` when busy
- Cells also have shortcuts called *magics*, for example
  - starting a code cell with `%%bash` or `!` allows us to run shell commands
  - `%cat` prints a file, `%load` loads a script, `%run` runs a script
  - `%set_env` and `%env` are used to set and list environment variables
- You can create a **Conda environment** and select it as the Notebook **kernel**
- Most common keyboard shortcuts:
  - **Shift+Enter**: run cell and jump to next cell
  - **Ctrl+/**: comment out or uncomment code

Lab: Getting comfortable with Jupyter Notebooks

> Start time: 9:15 am
>
> Expected completion time 10 minutes

Take 5 minutes to familiarize yourself with Jupyter Notebooks.

- Try creating a new Notebook
- Create new cells of different kind
- Run some cell magics
- Change the kernel
- Learn how to import existing Notebooks

Two Day Agenda

Day 1: Anomaly Detection in Structured Time Series Data

- Using AML SDK on Azure Databricks
  - Feature Engineering and Model Development w/ AML SDK
  - Hyperparameter Tuning with HyperDrive and AmlCompute (fka BatchAI)
  - Model Deployment with AKS

Day 2: Predictive Maintenance

- Using AML SDK on Datascience Virtual Machine (DSVM)
  - Feature Engineering and Model Development w/ AML SDK
  - Including Anomaly Detection for a multi-stage pipeline
  - Automated Machine Learning
  - Model Deployment with ACI

What are anomalies?

Any kind of data instance that is not normal.

Applications of anomaly detection

- intrusion detection
- fraud detection
- medical and public health anomaly detection
- industrial damage detection
- image processing
- anomaly detection in text data
- sensor networks

Intrusion detection

Host-based intrusion detection systems

> Anomalous subsequences (*collective anomalies*) in system call traces, caused by malicious programs, unauthorized behavior and policy violations.
> The *co-occurrence of events* that is the key factor in differentiating between normal and anomalous behavior.

Network intrusion detection systems

> Intrusions typically occur as anomalous patterns (*point anomalies*) though certain techniques model the data in a sequential fashion and detect anomalous subsequences (*collective anomalies*).

Fraud detection

Criminal activities occurring in commercial organizations

- banks (e.g. identity theft)
- credit card companies (e.g. fraudulent credit card applications or usage)
- insurance agencies (e.g. individual and conspiratorial rings of claimants and providers)
- cell phone companies (e.g. tethering)
- stock market (e.g. insider traiding)

Medical and public health anomaly detection

Anomalies can be due to several reasons: abnormal patient condition, instrumentation errors, or recording errors.

Unique Challenges:

> Privacy concerns arise when developing anomaly detection approach.
> Health risks to patients require extreme caution.

Industrial damage detection

Fault detection in mechanical units

> Monitor the performance of components such as motors, turbines, oil flow in pipelines, or other mechanical components, and detect defects that might occur due to wear and tear or other unforseen circumstances.

Structural defect detection

> Examples: cracks in beams or strains in airframes.

Other Applications

Image data

> Motion detection or abnormal regions in a static image (e.g. mammographic image analysis)

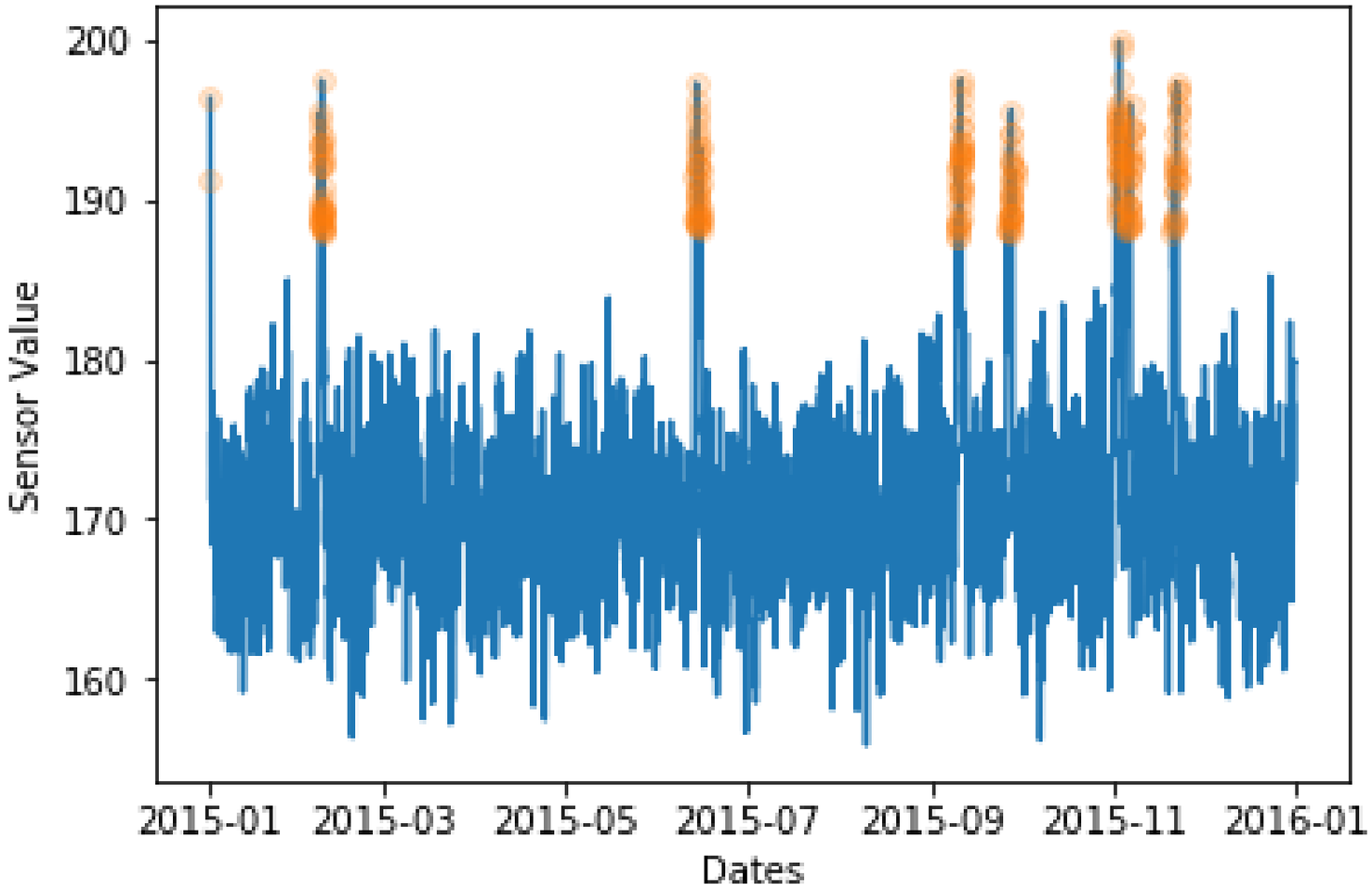
Text data

> Example: Detect novel topics, events, or news stories in a collection of documents or news articles.

Different kinds of anomalies

- Point anomalies
- Contextual anomalies
- Collective anomalies

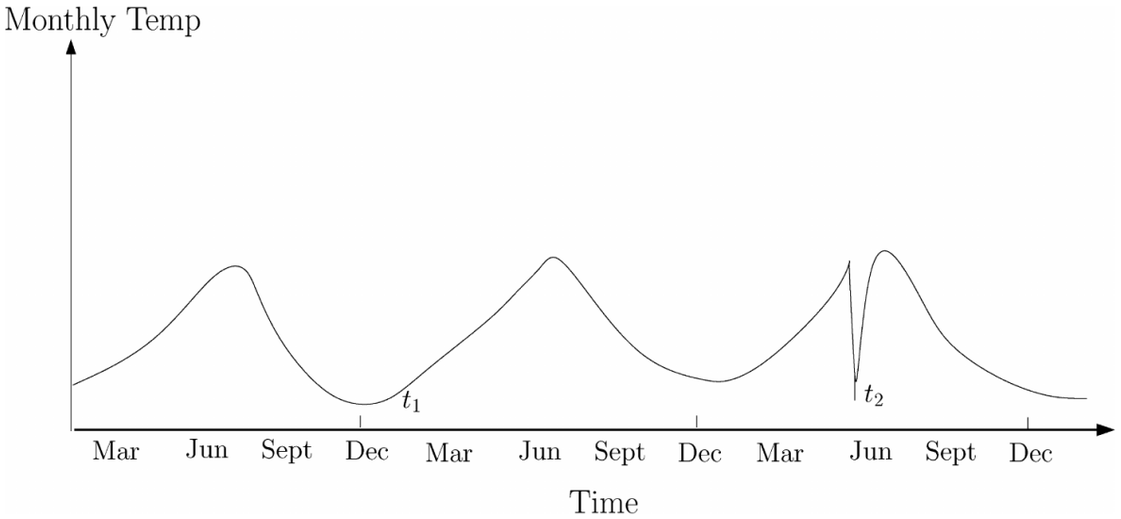
Point anomalies



One or more individual data instance can be anomalous with respect to the rest of data.

Contextual anomalies

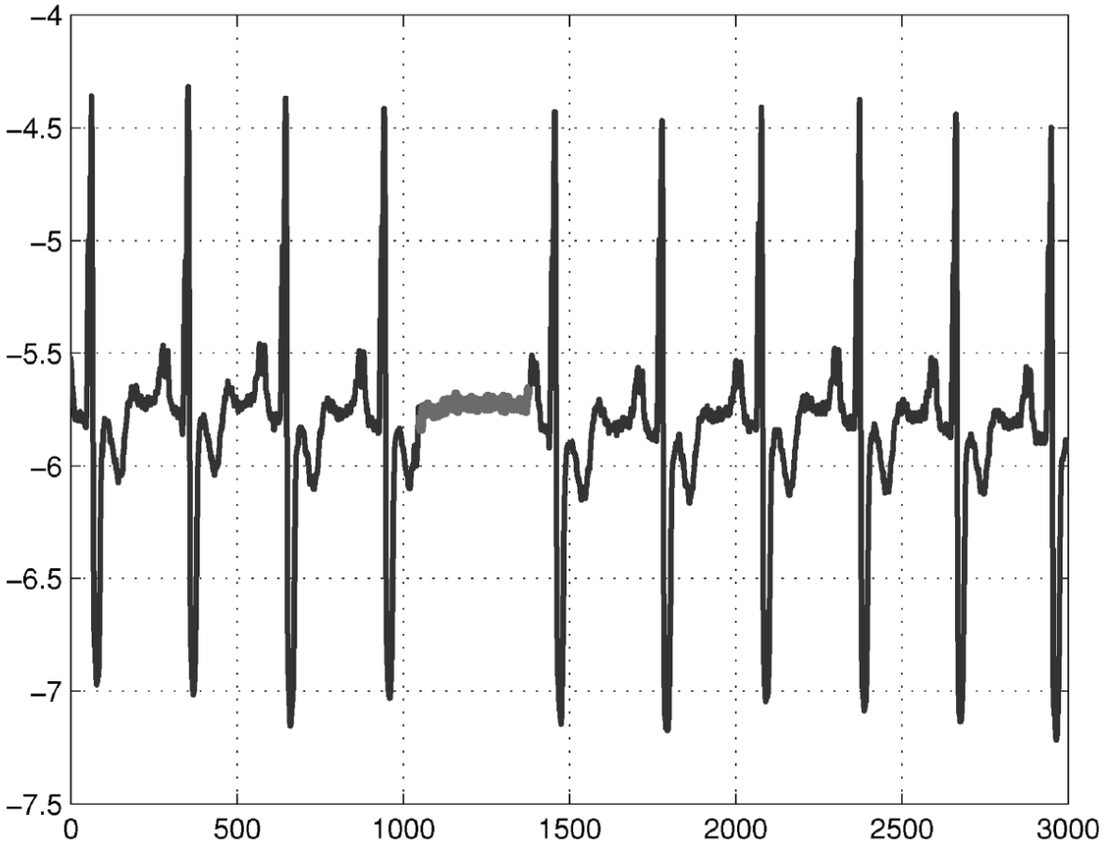
A data instance is *anomalous in a specific context*, but not otherwise.



> Example: Having an employee try to log onto her corporate account using her credentials is not anomalous per se. However, when this does not happen within the pre-defined business hours, the instance becomes anomalous in the temporal context.

Collective anomalies

The individual data instances in a collective anomaly may not be anomalies by themselves.



Classes of anomaly detection techniques

- Supervised anomaly detection
- Semisupervised anomaly detection
- Unsupervised anomaly detection

Supervised anomaly detection

Assumes the availability of a training data set that has labeled instances for normal as well as anomalous data

Two major issues:

- Anomalous instances are far fewer compared normal instances (imbalanced classes)
- Obtaining labels accurate and representative of future anomalies

Semisupervised anomaly detection

Assumes that the training data has labeled instances *only* for the normal class.

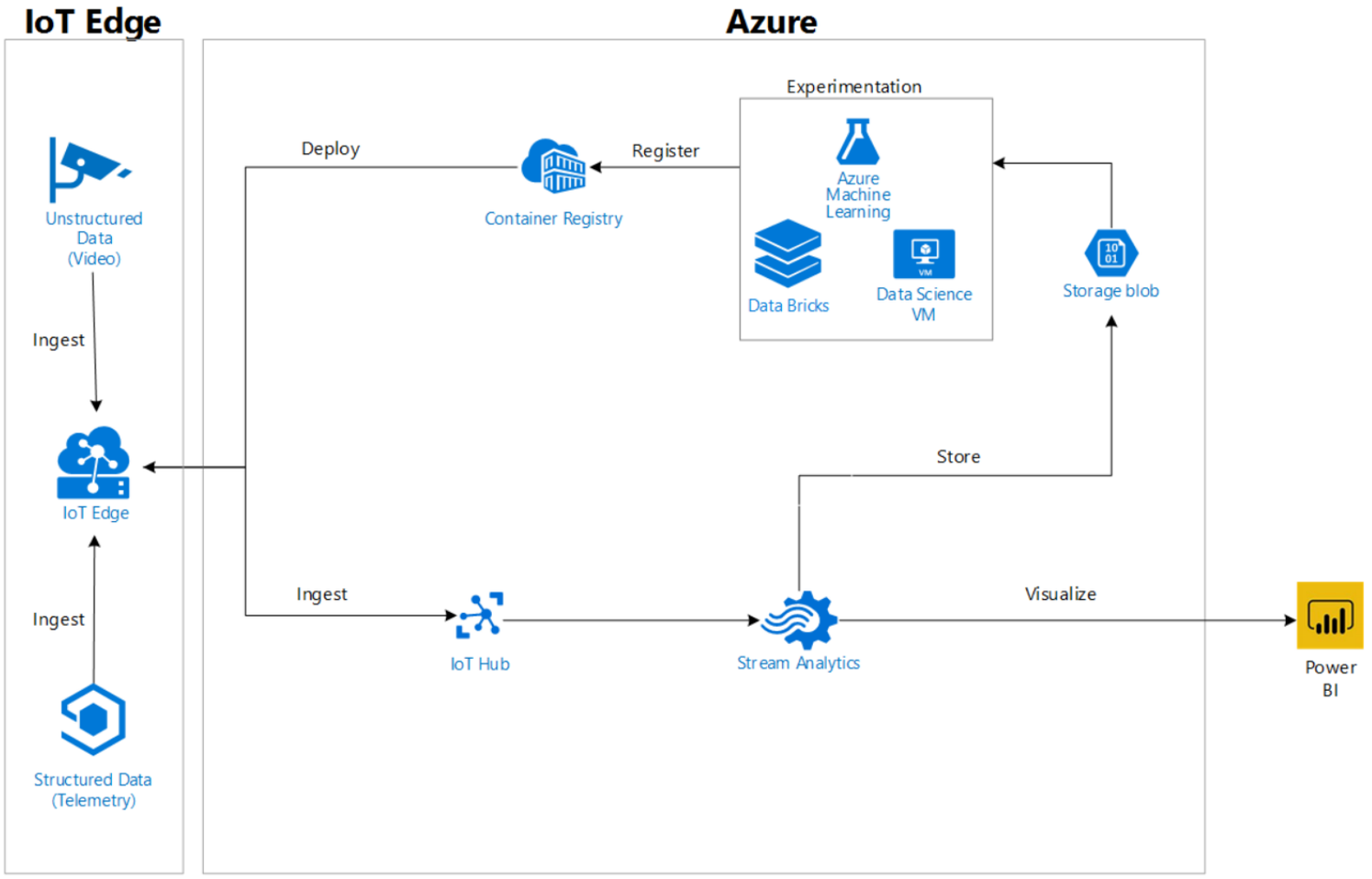
Typical approach:

> Build a model to predict normal behavior. If the predictions of the model fail to predict data instances, mark those instances as anomalies.

Unsupervised anomaly detection

Do not require training data, and thus are most widely applicable.

Final course architecture



Lab 1.1 (a): Setup and introduction to Anomaly Detection

Goal:

Develop an intuitive understanding of one of the most common and powerful approaches to Anomaly Detection in structured Timeseries data.

Outline:

1. Create a hypothetical time series dataset
2. Apply simple, but common anomaly detection
3. Understand the limitations of this approach in dealing with
    - seasonal trends (e.g. higher energy consumption on the weekends than on weekdays)
    - linear trends (e.g. growth over time)
4. Improve the anomaly detection algorithm to be robust against seasonal and linear trends

Open and run through lab 1.1 (a): Setup and introduction

> Start time: 9:40 am
>
> Expected completion time 20 minutes
>
> Feel free to work together
>
> There is a 10-minute break at the end
>
> Return at 10:10

Lab 1.1 (b): Data preparation for time series

Goals:

- Get to know your dataset better by visualizing it
- Learn how to visualize time series data
- Become familiar with a set of standard metrics that can be defined on time series data

Open and run through lab 1.1 (b): Data preparation for time series data

> Start time: 10 am
>
> Expected completion time 30 minutes
>
> Feel free to work together
>
> There is a 10-minute break at the end
>
> Return at 10:40

Lab 1.2: Model development

Goal:

- Learn how to integrate 3d-party open-source libraries into your project w/ AML SDK
- Use python port pyculiarity of Twitter's anomaly detection R library to detect anomalies in our telemetry data

Lab 1.2: Model development

Outline:

We will first develop a solution that processes all the data in *batch*, and then optimize the solution so that it detects anomalies *online*, as a new sensor measurement arrives.

Challenges:

1. How to calculate metrics online?
2. How many historic sensor readings do we need to consider to get robust anomaly detection?

Open and run through lab 1.2: Model development

> Start time: 10:40 am
>
> Expected completion time 1:20 minutes
>
> Feel free to work together
>
> There is a 1-hour lunch break at the end
>
> Return at 1 pm

Lab 1.3: Machine Learning Experimentation

Goals: Learn to perform ML Experimentation with the AML SDK:

- Create a high performance AMLCompute cluster for ML experimentation
- Log and investigate run history using the AML SDK and Azure Portal

Open and run through lab 1.3: Machine Learning Experimentation

> Start time: 1 pm
> Expected completion time 50 minutes
> Feel free to work together
> There is a 10-minute break at the end
> Return at 2:00 pm

Lab 1.3: Hyper Parameter Tuning w/ HyperDrive

Goal:

Learn how to use AML HyperDrive and Compute for optimizing your model

Open and run through lab 1.3: Hyper Parameter Tuning w/ HyperDrive

> Start time: 2 pm
> Expected completion time 50 minutes
> Feel free to work together
> There is a 10-minute break at the end
> Return at 3:00 pm

Lab 1.3: Model deployment

Goal:

Learn to deploy your machine learning solution as a AKS webservice w/ AML SDK

We will cover the following steps:

1. Recap: Configure your Azure Workspace.
2. Create a Docker image, using the Azure ML SDK.
3. Register your docker image in the Azure Container Registry (ACR).
4. Deploy your Docker image as an Azure Kubernetes Service (AKS) Webservice.
5. Test your Webservice.

Open and run through lab 1.3: Model Deployment

> Start time: 3 pm
>
> Expected completion time 60 minutes
>
> Feel free to work together

Q&A

End of Day 1

Further reading:

> - *Anomaly Detection: A Survey* by Varun Chandola, Arindam Banerjee, and Vipin Kumar
> - AML-SDK Documentation: https://docs.microsoft.com/en-gb/python/api/overview/azure/ml/intro?view=azure-ml-py)
> - AML-SDK Notebooks: https://github.com/Azure/MachineLearningNotebooks/

Agenda for course 2: Predictive maintenance

- Data exploration
- Training a model manually
- Training a model using automated ML
- Model deployment

The case for combining anomaly detection and predictive mainentance

Example use-case: *Anomalous* sensor measurements on industrial machines may be *predictive* of future catastrophic failure.

Solution:

> 1. Anomaly Detection - Detect anomalies in timeseries data
> 2. Predictive Maintenance - Predict future failures and schedule maintenance

Corrective maintenance (aka planned maintenance):

> Based on the assumption that a machine component will degrade within a given period of time that is common for its type. Maintenance is scheduled based on average or expected life statistics.

Predictive maintenance (aka preventive maintenance):

> By trying to predict failures before they happen and run just in time maintenance. This approach is often more optimal in balancing the cost failures and the labor and opportunity cost of equipment maintenance.

Common PdM use-cases

- **Airline:** Flight delay and cancellations due to mechanical problems
- **Banking:** ATM failure due to paper jam
- **Energy:** Wind turbine or circuit breat failures
- **Transportation:** Elevator door failures or train wheel failures

PdM data requirements

- static features (such as machine metadata)
- temporal or time-varying features (such as machine telemetries)
- error logs (anomalies or non-breaking events)
- maintenance or repair logs (breaking events requiring part replacement)
- operational history of the equipment including failures (labels)

PdM temporal data requirements

- data is collected at more or less regular time units
- time units are defined based on business justification
- time units don't have to be the same as the frequency of data collection
    - low frequency can mean there's not enough granurality in the data
    - high frequency means differences are smaller between adjacent time units

Considerations for training PdM models

- Choose between time-dependent split vs. splitting by machines/devices
- Carefully consider the important hyper-parameters and tune them
- Carefully consider model performance metrics based on misclassification cost
- Carefully consider various feature engineering approaches and the difficulty of implementing them at score time

Evaluating a PdM model

Precision

> Precision of PdM models relate to the rate of false alarms. Lower precision of the model generally corresponds to a higher rate of false alarms.

Recall

> Recall rate denotes how many of the failures were correctly identified by the model. Higher recall rates mean the model is successful in identifying the true failures.

F1 score

> F1 score is the harmonic average of precision and recall, with its value ranging between 0 (worst) to 1 (best). You can compute the F1 score based on the weights of your choosing.

PdM scoring

- To conform to the model signature, the features in the new data must be engineered in the same manner as the training data.
- Scenarios involving anomaly detection typically implement **online scoring** (aka **real-time scoring**).
- PdM usually involves batch scoring, but can be implemented in *near* real-time.

Open and run through lab 2.1

> Start time: 9:40
>
> Expected completion time 50 minutes
>
> Feel free to work together
>
> There is a 15-minute break at the end
>
> Return at 10:45

automated machine learning

- Searching for the right machine learning model can be very time-consuming
- You need to choose an algorithm and then choose the right hyper-parameters for it
- Automated ML does all that against a pre-defined compute target
- We need to be careful in our choice of the metric we optimize over
- Models selection should also be based on their expected performance in production
- Here's a list of algorithms that can be used for training
- You can configure your compute target

Open and run through lab 2.2

> Expected completion time 40 minutes
>
> Feel free to work together
>
> There is a 20-minute break at the end

Operationalization options

- We can deploy models as REST API end-points for online or batch scoring
- Deployment to ACI is quick and great for testing
- We specify the number of CPUs (1 by default) and gigabyte of RAM (1 by default) needed for your ACI container
- Deployment to AKS can be used for high-scale production scenarios
- Other deployment scenarios are available (see here)

Operationalization steps

- Register the model or retrieve a pre-registered model
  > The can be loaded from pickle file if not in current session or registered directly from the pickle file.
- Configure and create an image: provide execution script, Conda dependencies file, Python runtime
  - The execution script has an `init()` function for loading model and a `run()` function for getting predictions from data
  - Conda dependencies can be specified in a YAML config file
- Configure an ACI container: provide number of CPUs and RAM
- Deploy service into ACI container from image

Open and run through lab 2.3

> Expected completion time 45 minutes
>
> Feel free to work together
>
> There is a 15-minute break at the end

Debugging a live deployment

- We don't want to launch a new web service every time we change its base image
- We test our application by running the Docker container locally and making necessary updates
- We commit the changes in your Docker container to the Docker image and update the Azure Container Registry (ACR)
- Deploy your Docker image as an Azure Container Instance (ACI) Webservice

Open and run through lab 2.4

> Expected completion time 45 minutes
>
> Feel free to work together
>
> There is a 15-minute break at the end

Q&A

Please take the course survey

Please please pretty please

End of course 2

Further reading

> The predictive maintenance playbook

the End


Microsoft