VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Tussentijdse vergadering 16/05/2024

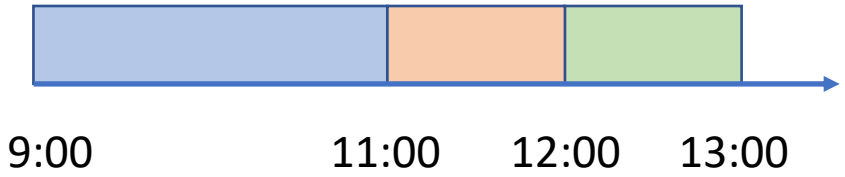
Locatie: Vandewiele nv

Met steun van VLAIO

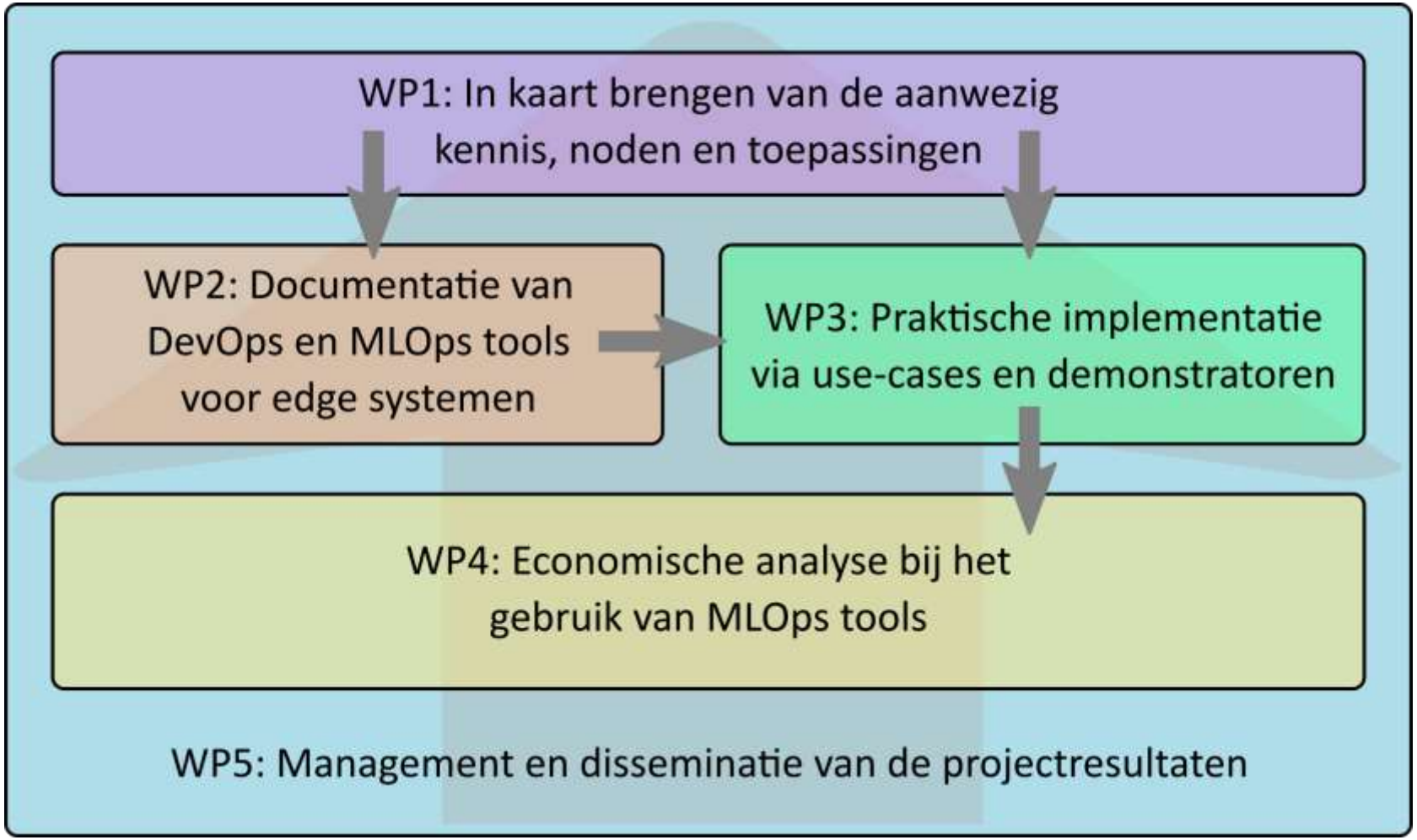
HBC.2023.0062

Agenda

- Introductie
- Overzicht MLOps Ecosysteem
- ONNX & Edge Deployment
- Cases en demonstratoren
- Volgende stappen
- Rondleiding Vandewiele nv
- Lunch & netwerken



Toelichting project - werkpakketten



Toelichting project - deliverables

- ✓ L1.1 Startvergadering
- ❑ L1.2 Rapport die de aanwezige kennis, mogelijke toepassingen beschrijft
- ✓ L2.1 Basiskennis verwerven rond DevOps en MLOps software tools die bruikbaar zijn op edge systemen
- ❑ L2.2 Handleiding met voorbeelden die de toepassing en werking van de software tools binnen MLOps voor edge devices bundelt
- ❑ L3.1 Uitwerking van de vijf bedrijfscases aangeleverd door de begeleidingsgroep
- ❑ L3.2 Inschakelen van studenten voor het uitwerken van de use-cases en demonstratoren
- ❑ L3.3 Rapport over de vijf uitgewerkte use-cases
- ❑ L3.4 Ontwikkeling van twee academische demonstratoren
- ❑ L4.1 Haalbaarheidsstudie voor de implementatie van de projectresultaten van elke case
- ❑ L4.2 Rapport over de economische evaluatie
- ❑ L5.1 Cursusmateriaal dat zal ingezet worden binnen de vakken
- ❑ L5.2 Organisatie van een seminarie om de kennis van de tools van MLOps voor edge systemen over te brengen
- ❑ L6.3 Finale begeleidingsgroep vergadering
- ❑ L5.4 Website
- ❑ L5.5 Verslaggeving naar alle stakeholders

Update begeleidingsgroep

- Beckhoff
- Bekaert*
- CNHi
- CTRL Engineering
- **Leap Technologies**
- LET
- Marelec
- Radix
- Siemens
- **Summa nv**
- Televic
- Vandewiele
- Vintecc
- Yazzoom
- Odisee (waarnemer)
- Thomas More (waarnemer)
- Howest (waarnemer)

***TODO: Reglement van Orde (RvO)**

Feedback vorige bevraging

Responsgraad: 60%

Is het project nog relevant voor de onderneming/organisatie:	4.67/5
Projectverloop en voorlopige resultaten voldoende:	4.11/5
Tevredenheid over ruimte voor overleg en sturing in project:	4.56/5
Tevredenheid van de behandelde punten in de vergadering:	4.22/5
Verwachting van de toepassing van de resultaten in bedrijf:	3.89/5

Feedback: algemeen positief

Overzicht MLOps Ecosysteem

Lara Luys

MLOps motivation

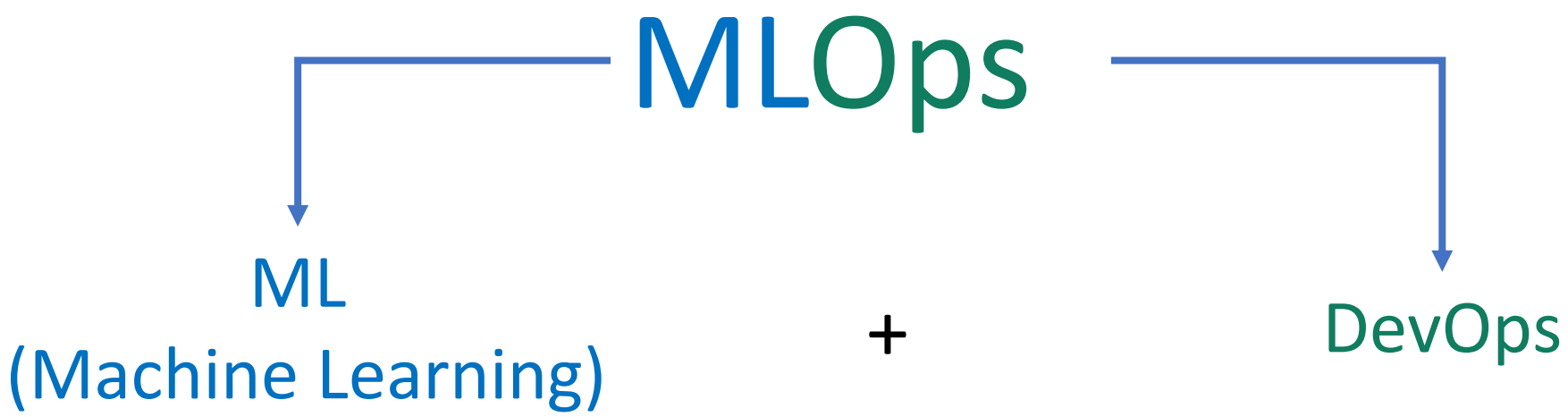
Before any machine learning model can be put in production,
many experimentation cycles are needed to identify the right ML model
to achieve the business goal.

3 main **artefacts**:

DATA + MODEL + CODE

➔ Need for well-defined structures, processes, and proper software tools
that manage these artifacts over the machine learning cycle!

MLOps

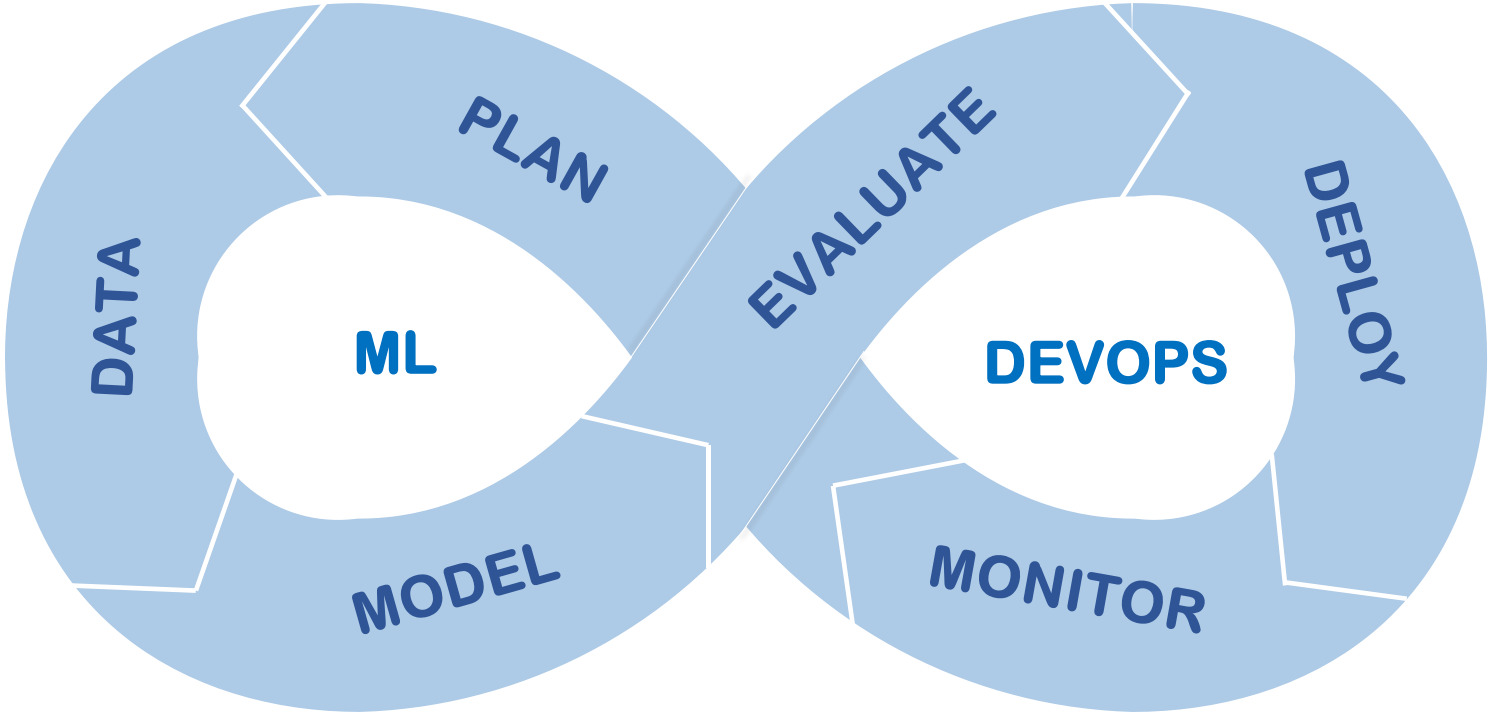


Paradigm used to **create** and **maintain** a machine learning model that will be deployed in **production**

DevOps vs. MLOps

	DevOps	MLOps
Code versioning	☑	☑
Compute environment	☑	☑
Continuous integration/delivery (CI/CD)	☑	☑
Monitoring in production	☑	☑
Data provenance		☑
Datasets		☑
Models		☑
Hyperparameters		☑
Metrics		☑
Workflows		☑

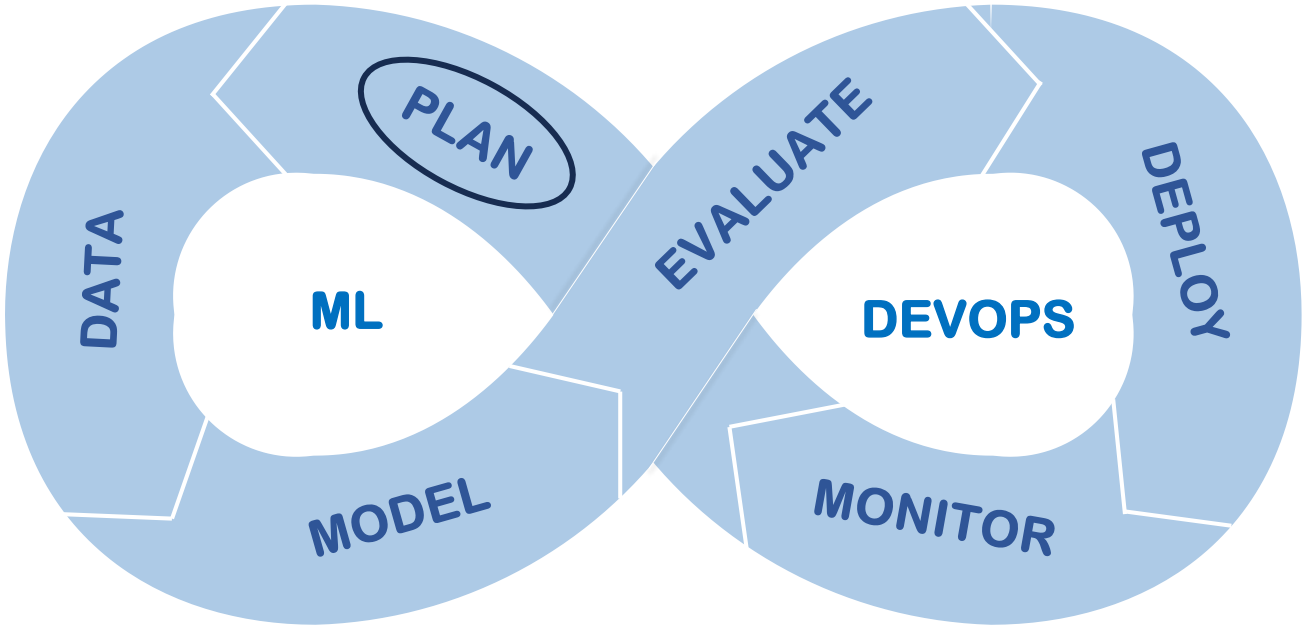
MLOps pipeline



A Lot of tools …

The pipeline

- **Plan**
- Data
- Model
- Evaluate
- Deploy
- Monitor
- Closing the loop



Plan – what

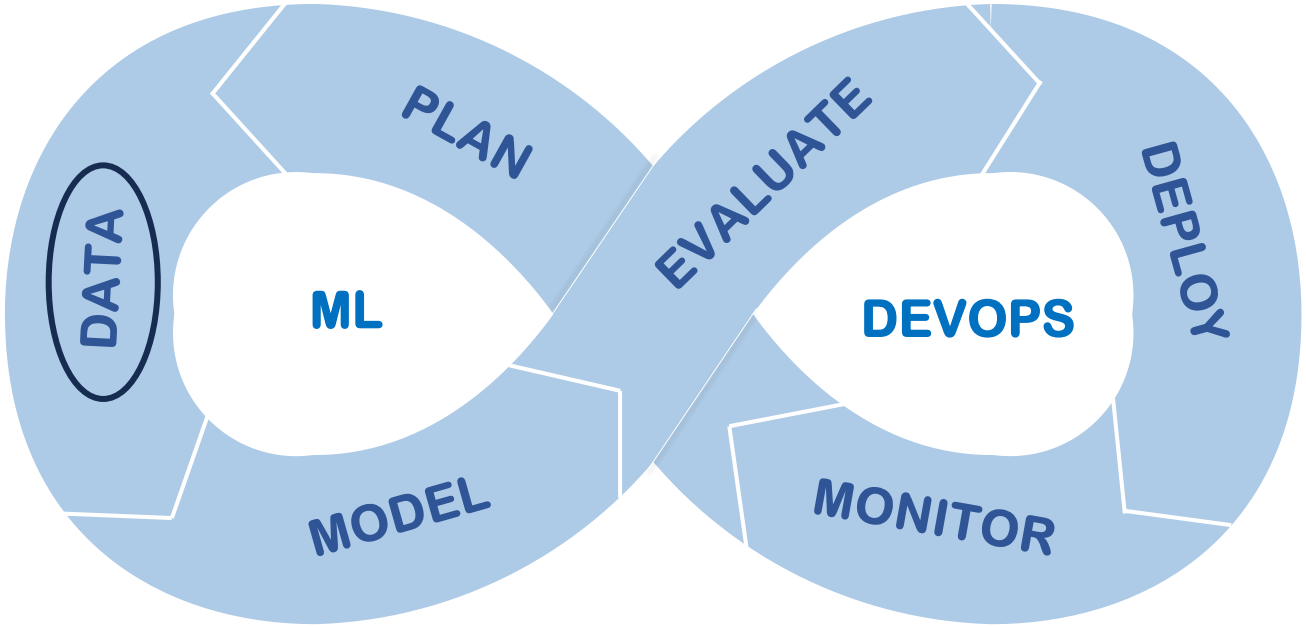
- Identify the problem together with all parties
- Plan a solution:
  - What does the solution need to do?
  - Which data needs to be collected?
  - Which type of model will be used?
  - Establish a workflow orchestration platform
  - Setup code version control
  - Decide on the used tools
  - …

Plan – tools

- Version control: Github, Gitlab, …
- Workflow orchestration (pipeline tools): Prefect, Airflow, Argo …

The pipeline

- Plan
- **Data**
- Model
- Evaluate
- Deploy
- Monitor
- Closing the loop



Data – what

- Collect the data
- Data labelling
- Data preprocessing
- Create data versioning

Data – data labelling tools

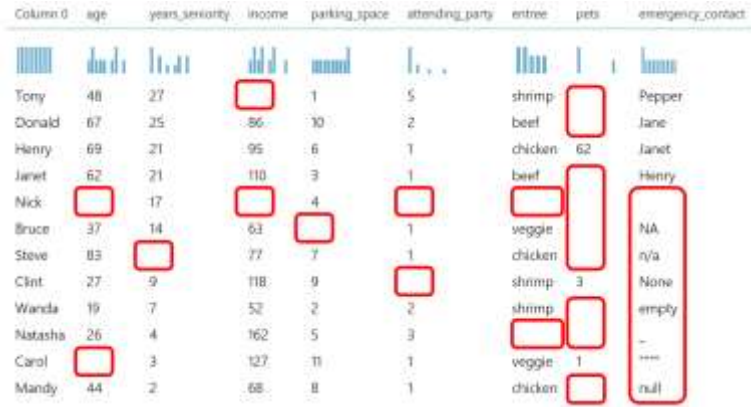
- Data labeling can take a lot of time
- Use tools to make labeling easier
  - Vision
  - Sound
  - Text
  - Self made
- E.g., Label studio, Prodigy, LabelBox, time series annotator …

Prodi.gy

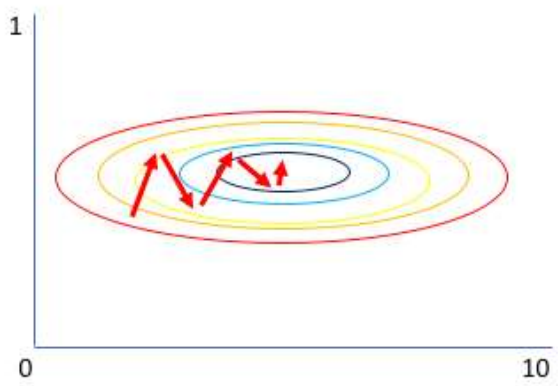
Time series annotator

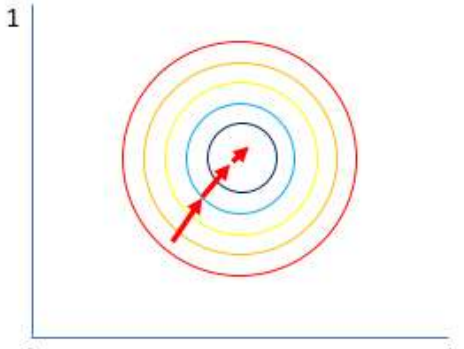
Data – data preprocessing

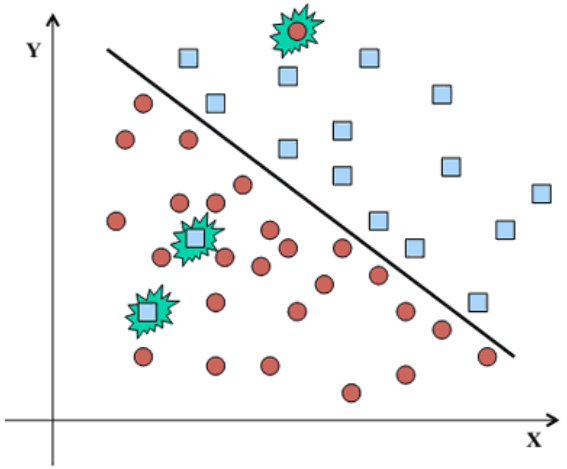
- Data cleaning: missing data, noisy data
- Data transformations: normalization
- Data reduction
- Data augmentation
- Data splitting



Column 0	age	years_seniority	income	parking_space	attending_party	entree	pets	emergency_contact
Tony	48	27		1	5	shrimp		Pepper
Donald	67	25	86	10	2	beef		Jane
Henry	69	21	95	6	1	chicken	62	Janet
Janet	62	21	110	3	1	beef		Henry
Nick		17		4				
Bruce	37	14	63		1	veggie		NA
Steve	83		77	7	1	chicken		n/a
Clint	27	9	118	9		shrimp	3	None
Wanda	19	7	52	2	2	shrimp		empty
Natasha	26	4	162	5	3			-
Carol		3	127	11	1	veggie	1	****
Mandy	44	2	68	8	1	chicken		null







Data – versioning tools

- DVC (data version control), LakeFS, Delta Lake, Pachyderm...

Data – DVC

- Data Version Control
- Open-source data versioning tool
- Easy to use + lightweight
- Extension of Git with simple command line commands
- Has functionality to manage pipelines and ML models

Data – demo

Demo: DVC

Demo – Water Quality

- Inputs: water parameters
- Output: potability (yes/no)
- Dataset: https://www.kaggle.com/datasets/adityakadiwal/water-potability

Dataset

- 9 input features
- 1 output feature
- Data cleaning:
  - Remove NaN
  - Normalization

```
data shape: (3276, 10)

            ph    Hardness        Solids  Chloramines     Sulfate  \
0          NaN  204.890455  20791.318981     7.300212  368.516441
1     3.716080  129.422921  18630.057858     6.635246         NaN
2     8.099124  224.236259  19909.541732     9.275884         NaN
3     8.316766  214.373394  22018.417441     8.059332  356.886136
4     9.092223  181.101509  17978.986339     6.546600  310.135738
...        ...         ...           ...          ...         ...
3271  4.668102  193.681735  47580.991603     7.166639  359.948574
3272  7.808856  193.553212  17329.802160     8.061362         NaN
3273  9.419510  175.762646  33155.578218     7.350233         NaN
3274  5.126763  230.603758  11983.869376     6.303357         NaN
3275  7.874671  195.102299  17404.177061     7.509306         NaN

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
0       564.308654       10.379783        86.990970   2.963135           0
1       592.885359       15.180013        56.329076   4.500656           0
2       418.606213       16.868637        66.420093   3.055934           0
3       363.266516       18.436524       100.341674   4.628771           0
4       398.410813       11.558279        31.997993   4.075075           0
...            ...             ...              ...        ...         ...
3271    526.424171       13.894419        66.687695   4.435821           1
3272    392.449580       19.903225              NaN   2.798243           1
3273    432.044783       11.039070        69.845400   3.298875           1
3274    402.883113       11.168946        77.488213   4.708658           1
3275    327.459760       16.140368        78.698446   2.309149           1
```

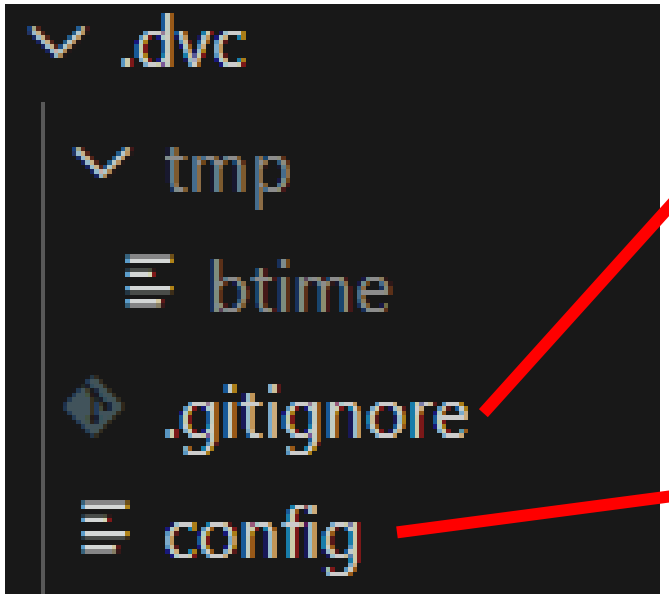
DVC - setup

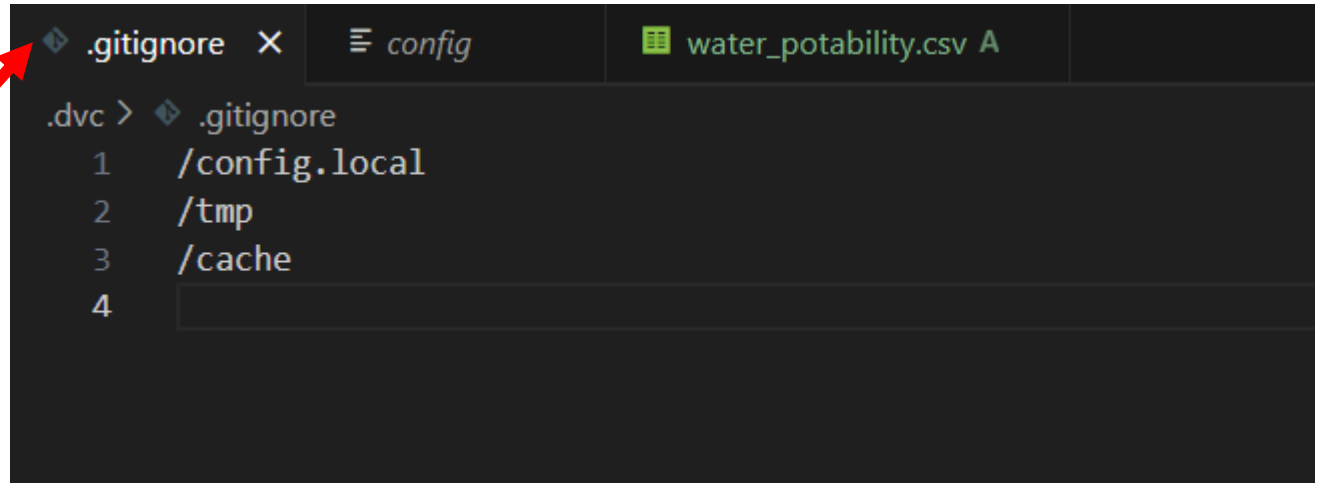
- Create git repository
- Create dvc repository

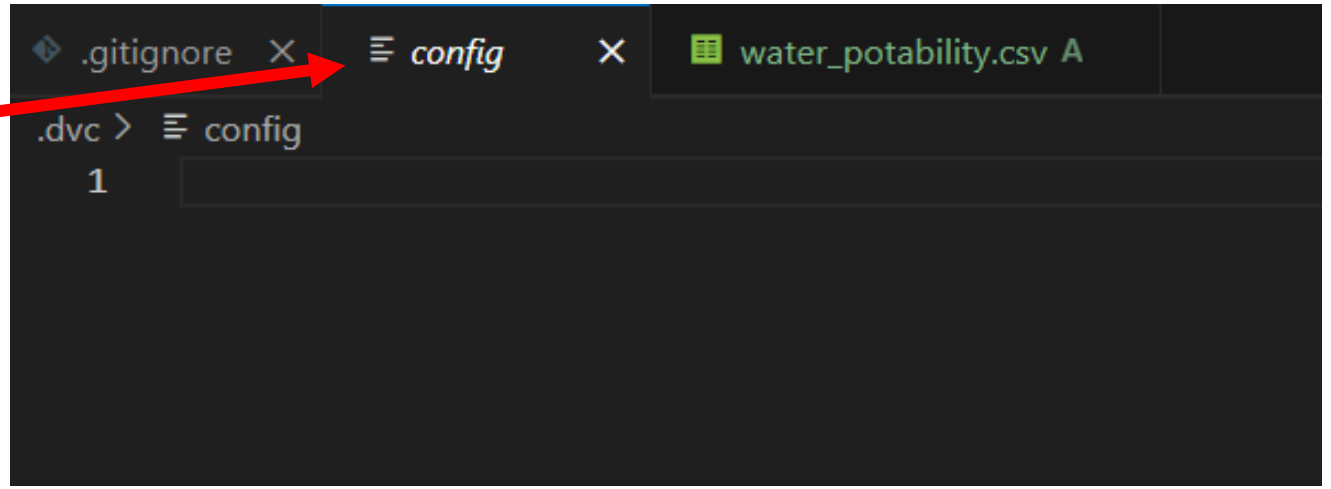
```
PS C:\Users\laral\Visual_studio_code\mlops-demo> dvc init
Initialized DVC repository.

You can now commit the changes to git.
```

- Creates several files

DVC - added files







DVC - add data repository locally

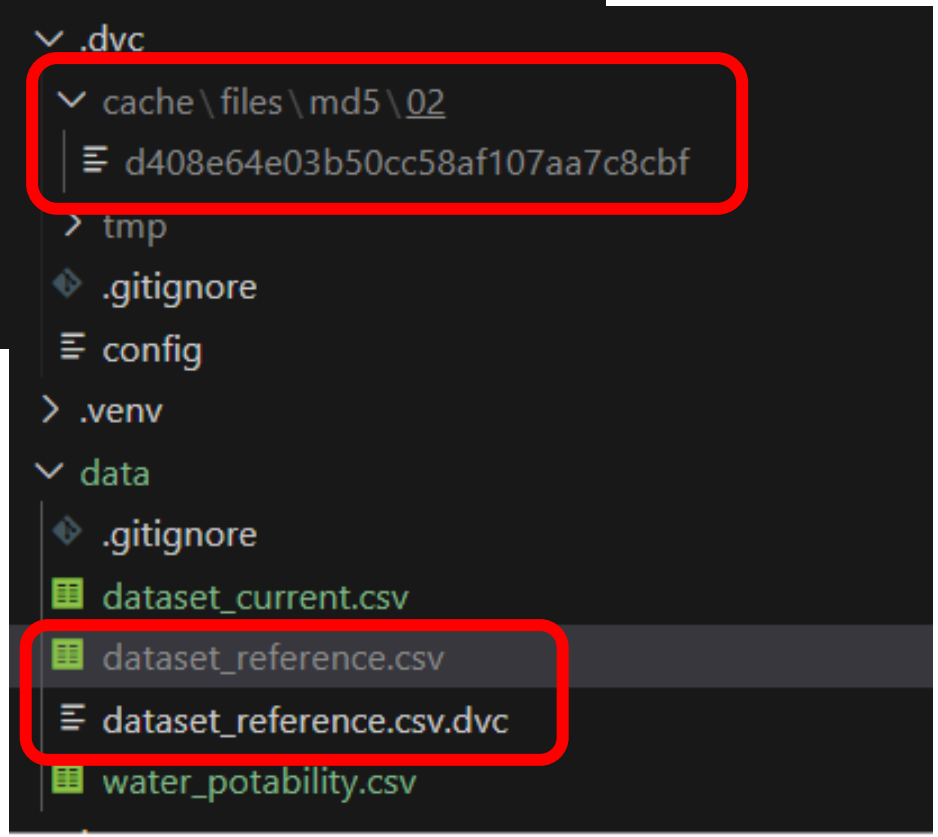
.gitignore | dataset_reference.csv

data > dataset_reference.csv > data

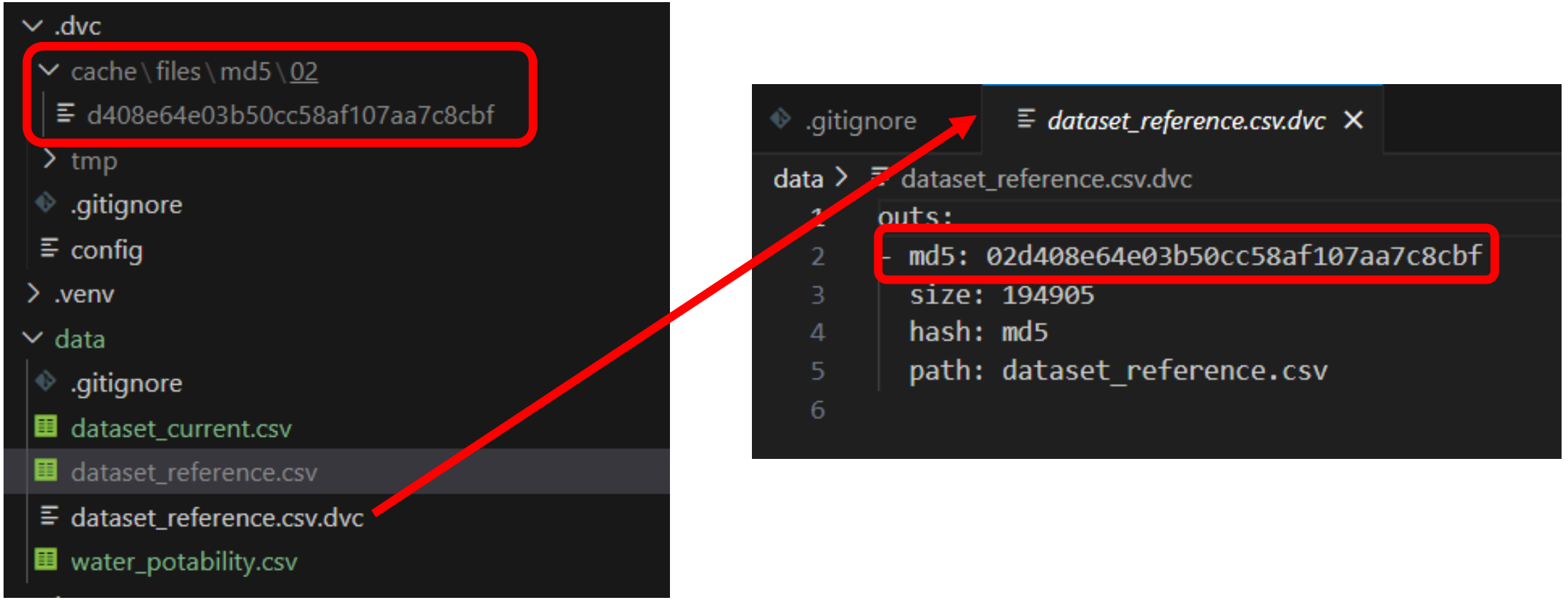
```
1    ,ph,Hardness,Solids,Chloramines,Sulfate,Conductivity,Organic_carbon,Trihalomethanes,Turbidity,Potability
1168 3204,8.077260885586492,125.30271940363592,23931.28283313192,8.773161930761969,317.6933311873576,398.32878874006775,15.2
1169 3207,9.24141997355796,127.91882621440118,39566.75435226551,8.860818357278797,281.9956396347743,487.3391692888744,9.5344
1170 3212,5.913755319868532,175.3260618694199,12044.624691463148,8.368785486252278,347.88037185456227,380.9671658106569,12.5
1171 3217,,98.3679148956603,28415.57583214058,10.55894999846796,296.843207792478,505.24026927891407,12.882614472289331,85.32
1172 3225,8.468741062907595,132.45648354840728,21038.442612673443,8.001377824874378,345.00073466919486,360.5755864381471,10.
1173 3230,8.659113026830287,114.8075783848566,23514.63664741336,8.735315343467212,333.0272047391256,318.64067926297173,16.55
1174 3233,10.485603647205735,136.57738115579482,32872.38056577991,8.399435299469037,276.92183480609395,416.1885459071077,18.
1175 3238,9.186453377063309,169.1198051240283,27055.69626088422,7.442479684166754,,498.6353223340039,10.247444234540184,50.4
1176 3239,9.960689857533172,169.92524845007168,15835.122817712849,7.143906655908775,301.77725263475696,388.8971587772877,13.
1177 3246,10.667363934157889,173.381945364966,28912.202201016768,7.071294493360232,276.63439102605554,286.063394408431,17.68
1178 3247,8.628301074066663,185.9267230581658,31548.00646369594,7.079462297434495,,342.35569745032865,18.248367893304533,62.
1179 3250,7.371914155906743,148.1936975614508,42059.38041717759,7.966710438575026,324.54626205550505,544.8484318253602,17.16
1180 3254,8.862112709633287,131.63517654759482,17433.60185279686,7.639572886949698,340.13316533016723,399.4628435734118,16.7
1181 3256,7.607223911235492,160.5652530477201,39184.8467196431,7.8264110490998995,312.0560664594202,503.1580785347783,13.366
1182 3258,6.638411449233408,180.8266674369293,9772.504814179669,8.295983092416883,,401.1111433997851,12.601517331439425,61.0
1183 3259,9.271355446767778,181.25961724022244,16540.97904800414,7.022499178514222,309.2388651146156,487.69278784471584,13.2
1184 3260,,134.73685568990112,9000.025591189238,9.026292722975022,,428.21398699299374,8.668672182023442,74.7733924065712,3.6
1185 3266,8.37291028476714,169.08705216090655,14622.745494434115,7.54798401839384,,464.52555235732217,11.083026574437897,38.
1186 3269,11.491010908391427,94.81254521783356,37188.826021934,9.263166160944529,258.930600411521,439.8936179168196,16.17275
1187 3273,9.41951031641321,175.76264629629543,33155.578218312294,7.350233233214412,,432.0447830453679,11.039069688154314,69.
1188
```

DVC – add data to DVC

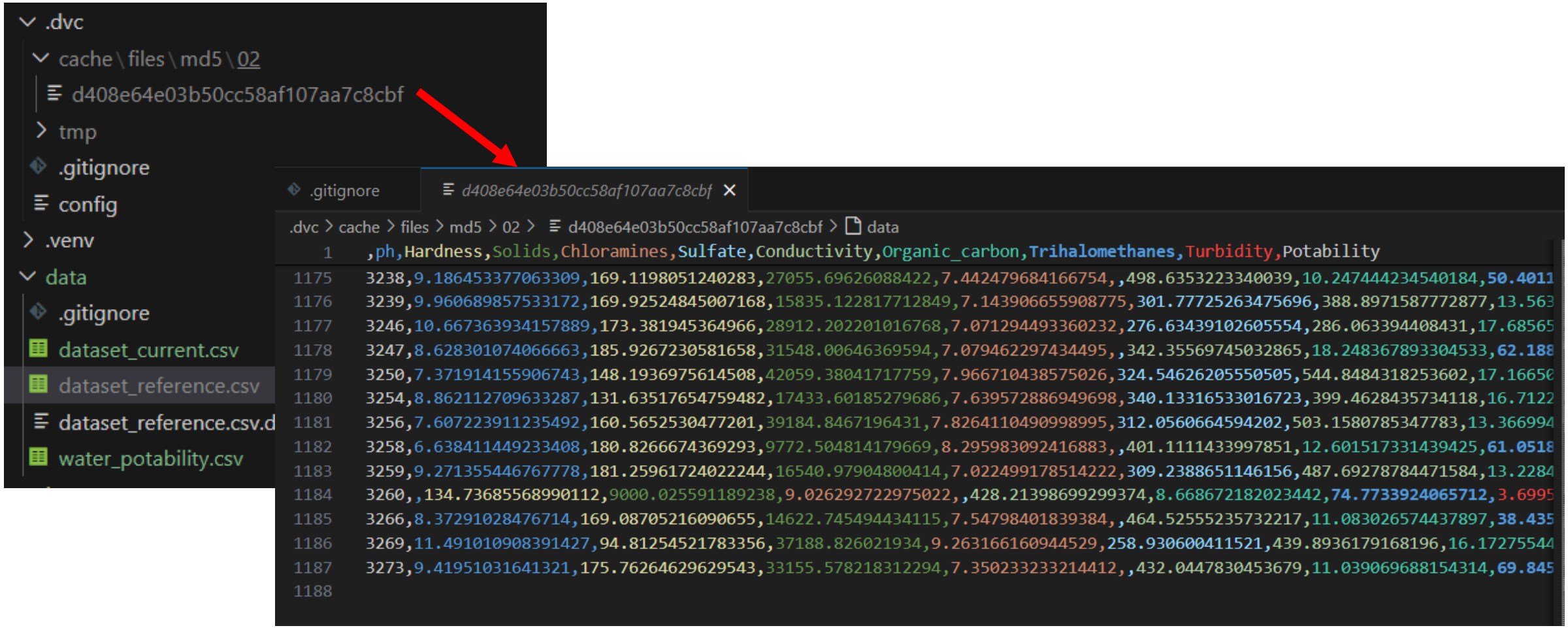
```
PS C:\Users\laral\Visual studio code\mlops-demo> DVC add data/dataset_reference.csv
100% Adding...|

To track the changes with git, run:

        git add 'data\dataset_reference.csv.dvc'

To enable auto staging, run:

        dvc config core.autostage true
```



DVC – new files explained



DVC – new files explained

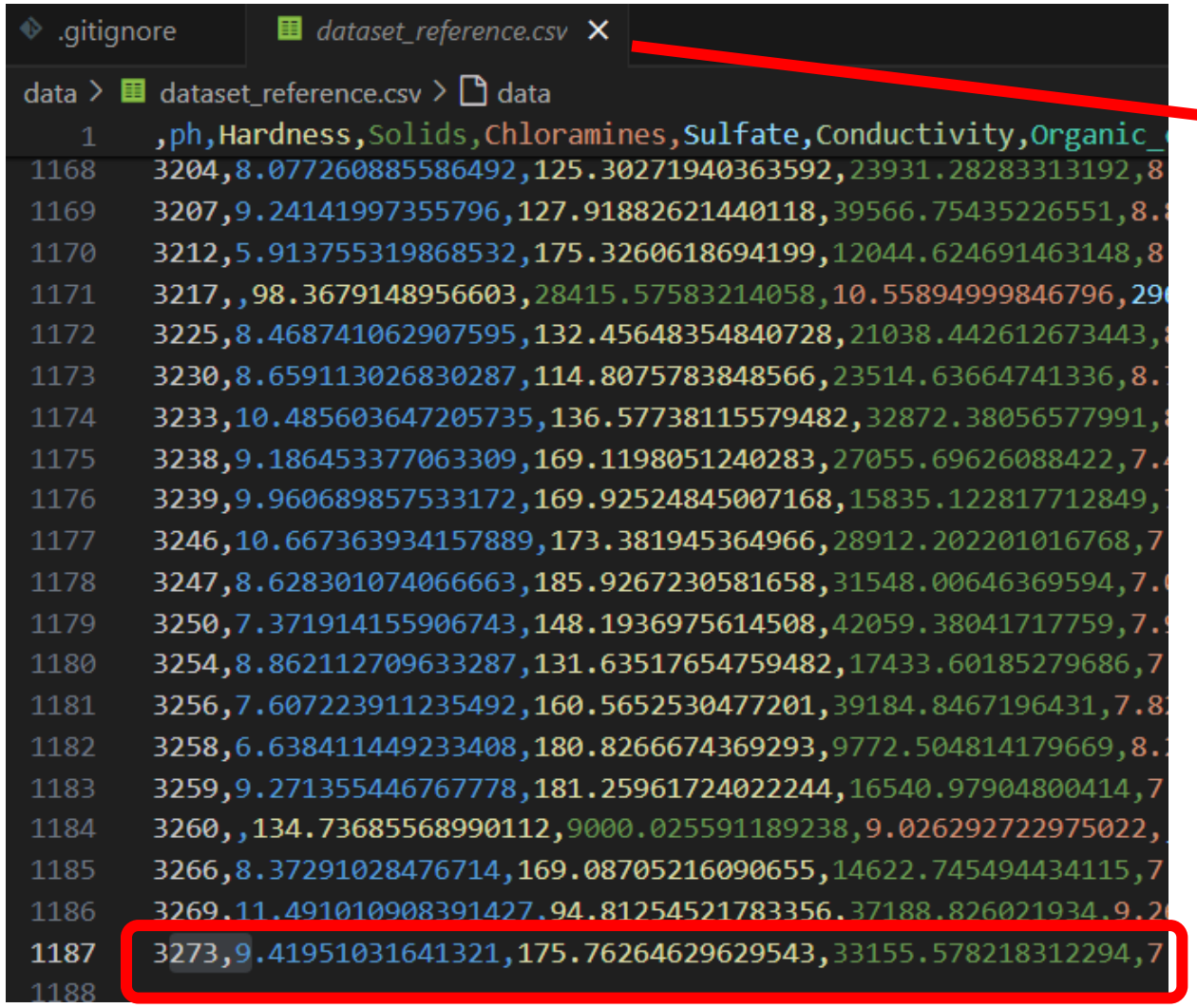


DVC – add DVC to Git

```
PS C:\Users\laral\Visual_studio_code\mlops-demo> DVC add data/dataset_r
100% Adding...|

To track the changes with git, run:

        git add 'data\dataset_reference.csv.dvc'

To enable auto staging, run:

        dvc config core.autostage true
```

```
PS C:\Users\laral\Visual_studio_code\mlops-demo> git add data/dataset_reference.csv.dvc
PS C:\Users\laral\Visual_studio_code\mlops-demo> git commit -m "added dataset_reference"
[detached HEAD 4cdc67a] added dataset_reference
 7 files changed, 5471 insertions(+), 54 deletions(-)
 create mode 100644 .prefectignore
 create mode 100644 data/dataset_current.csv
 create mode 100644 data/water_potability.csv
 create mode 100644 prefect.yaml
PS C:\Users\laral\Visual_studio_code\mlops-demo>
```

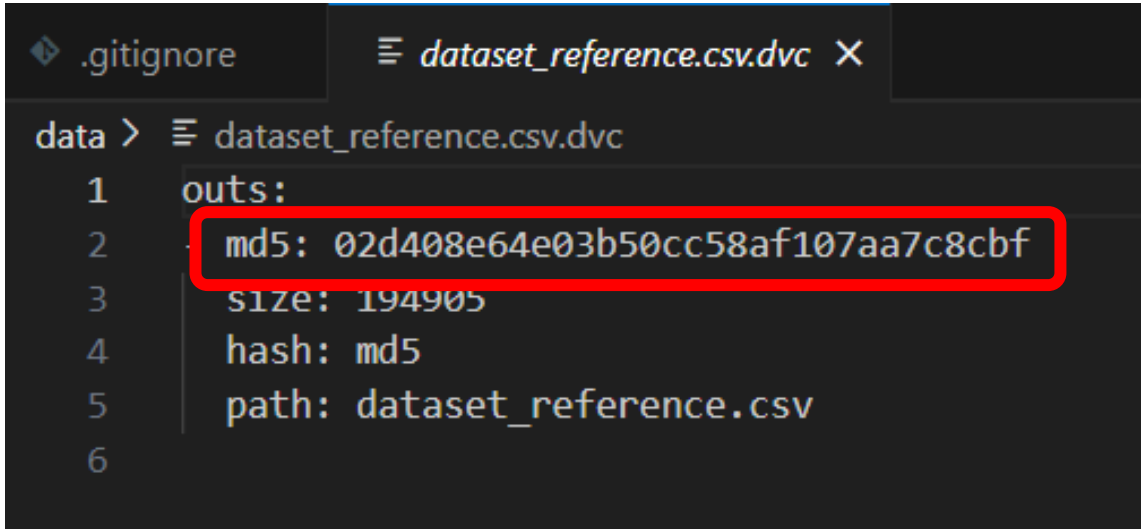
DVC – change file

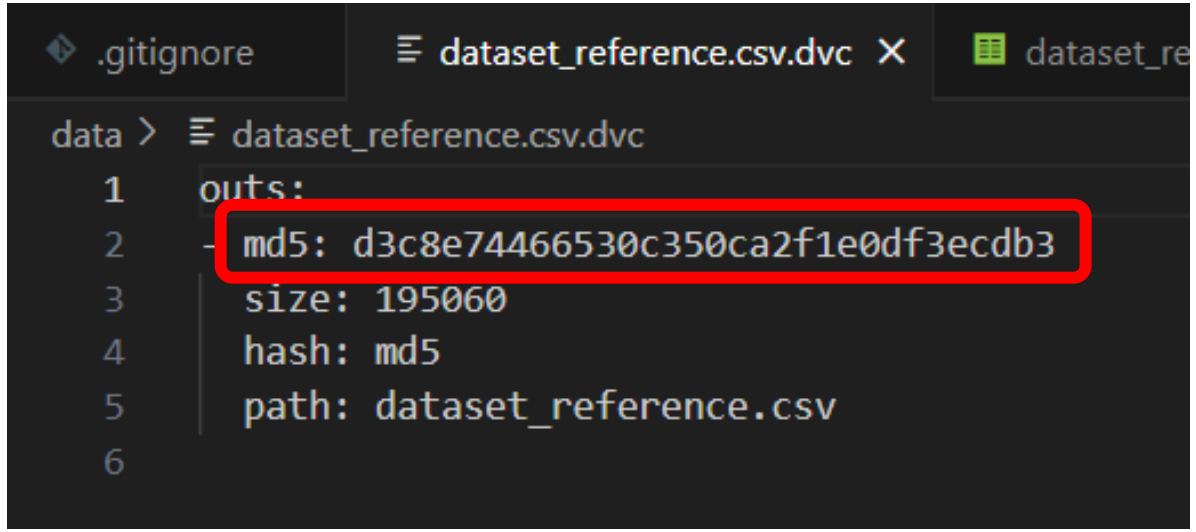


DVC – add changes

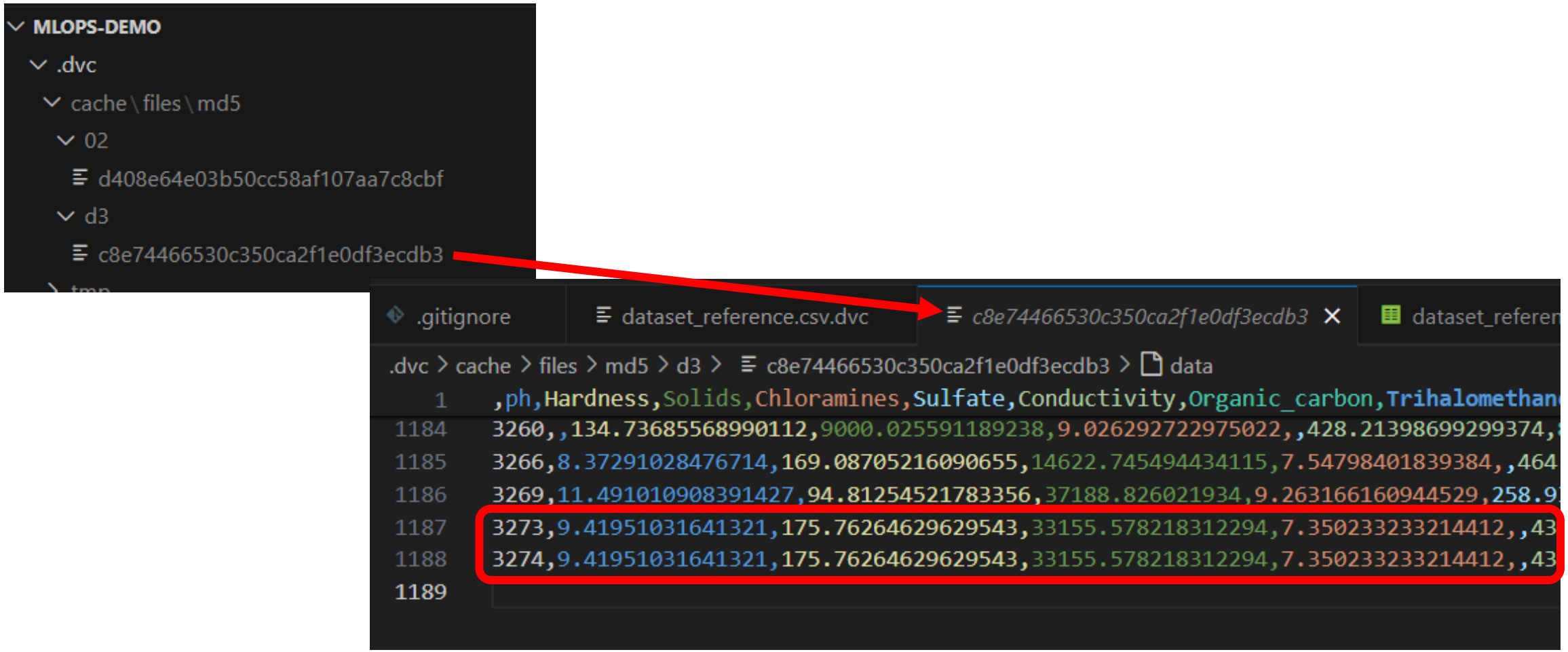
```
PS C:\Users\laral\Visual_studio_code\mlops-demo> DVC add data/dataset_reference.csv
100% Adding...|                                                                                |1/1 [00:00,  9.67file/s]

To track the changes with git, run:

        git add 'data\dataset_reference.csv.dvc'

To enable auto staging, run:

        dvc config core.autostage true
PS C:\Users\laral\Visual_studio_code\mlops-demo> git add data/dataset_reference.csv.dvc
PS C:\Users\laral\Visual_studio_code\mlops-demo> git commit -m "changed dataset_reference"
[detached HEAD ffb3838] changed dataset_reference
 1 file changed, 2 insertions(+), 2 deletions(-)
PS C:\Users\laral\Visual_studio_code\mlops-demo>
```

DVC – show what changes did





DVC – show changes

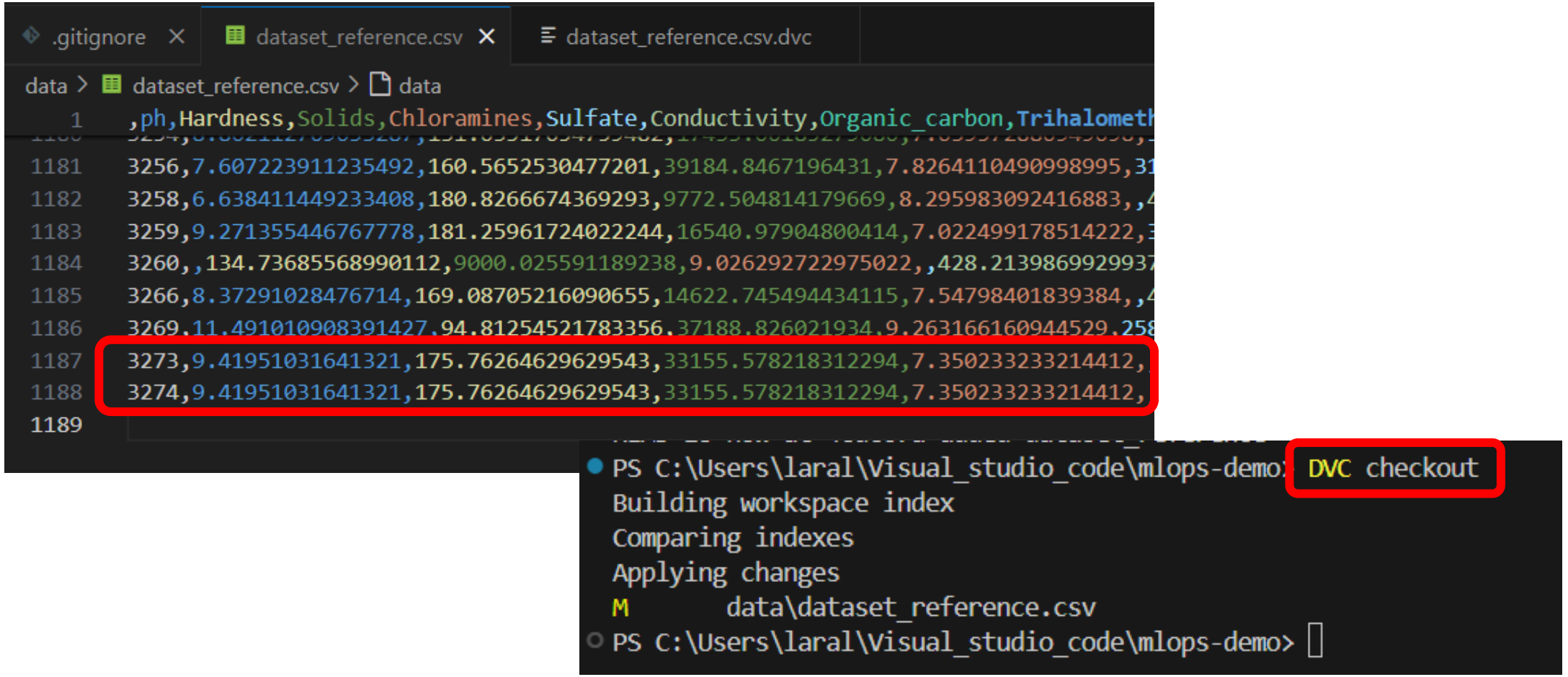


DVC – checkout earlier version

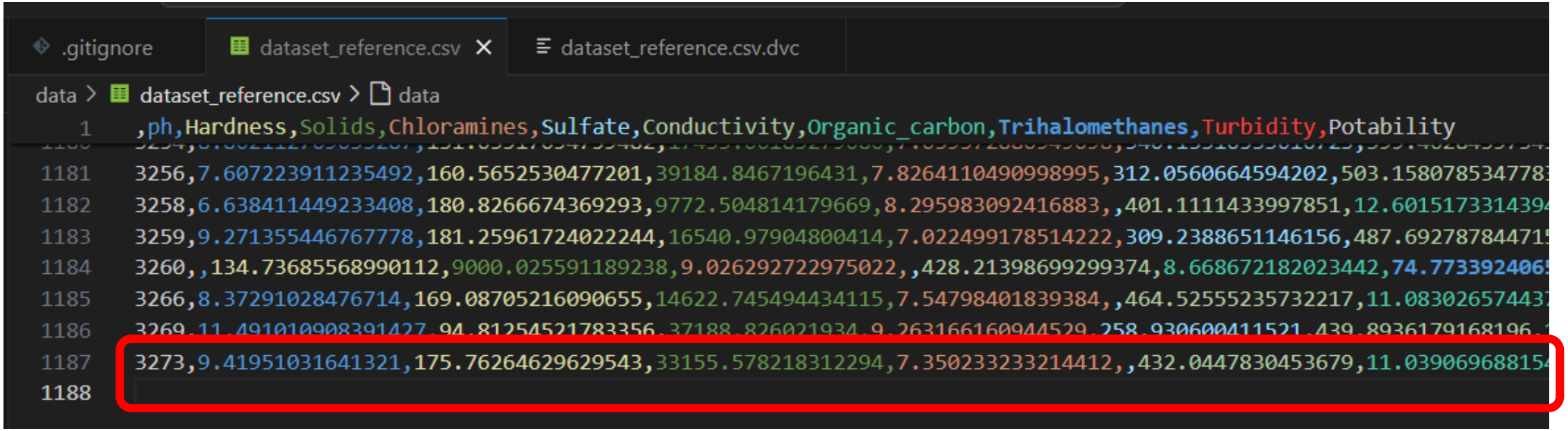
```
PS C:\Users\laral\Visual_studio_code\mlops-demo> git log
commit ffb3838ad5315c192ab10e869a3d91d8312e6c52 (HEAD)
Author: LaraLuys <lara.luys@gmail.com>
Date:   Mon May 13 15:24:23 2024 +0200

    changed dataset_reference

commit 4cdc67ae2d2b04c3d8bd2d2287480cb0720ba1df
Author: LaraLuys <lara.luys@gmail.com>
Date:   Mon May 13 15:18:18 2024 +0200

    added dataset_reference
```

```
PS C:\Users\laral\Visual_studio_code\mlops-demo> git checkout 4cdc67ae2d2b04c3d8bd2d2287480cb0720ba1df
```

DVC – checkout earlier version

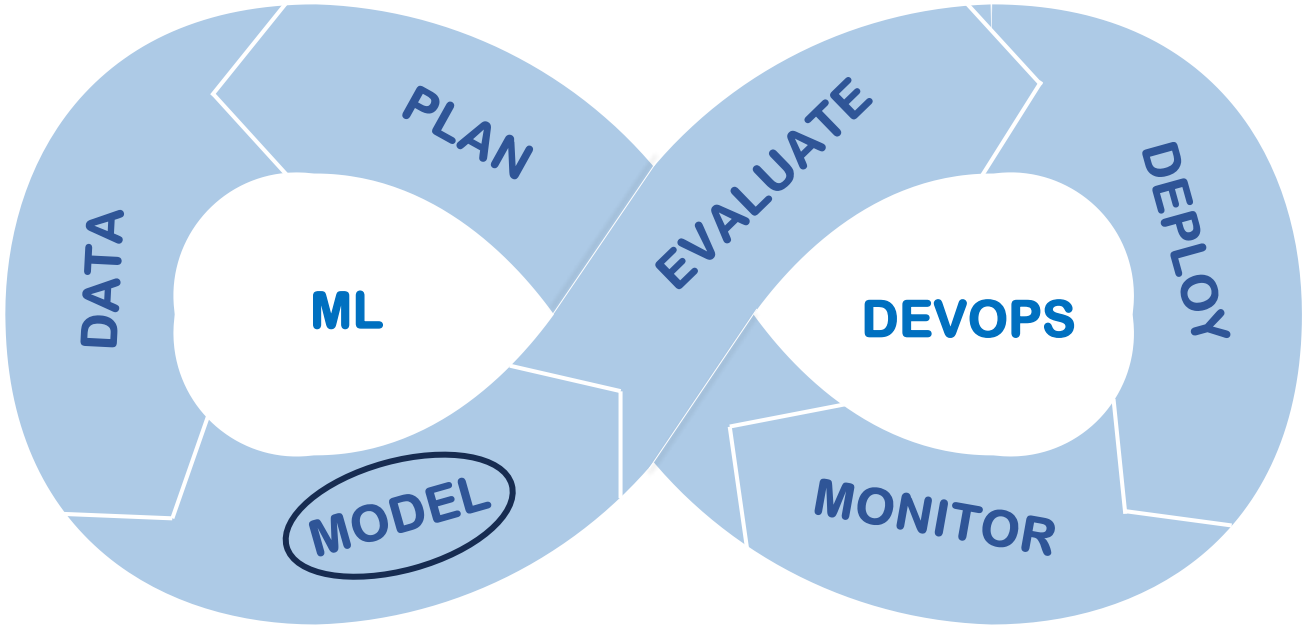


DVC – checkout earlier version



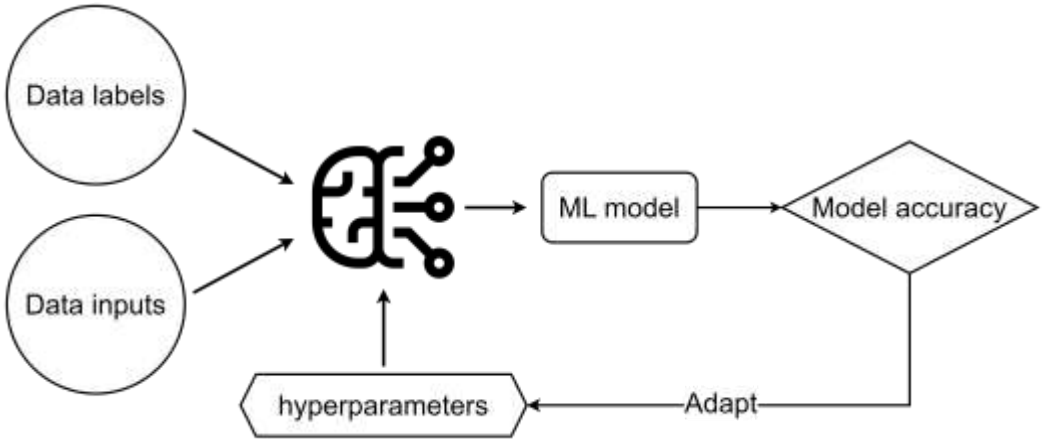
The pipeline

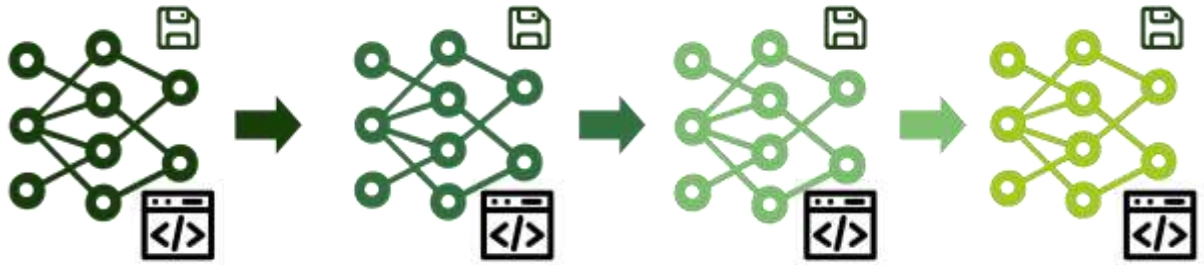
- Plan
- Data
- **Model**
- Evaluate
- Deploy
- Monitor
- Closing the loop

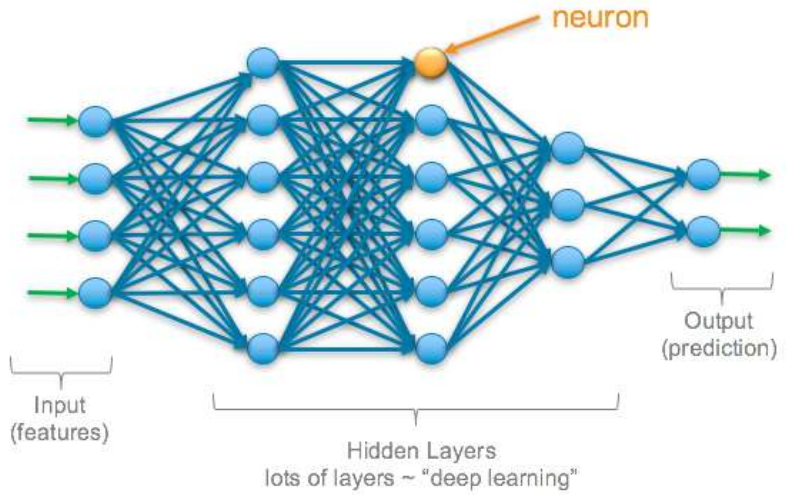


Model – what

- Model architecture creation
- Model training
- Model finetuning
- Create model + metadata versioning





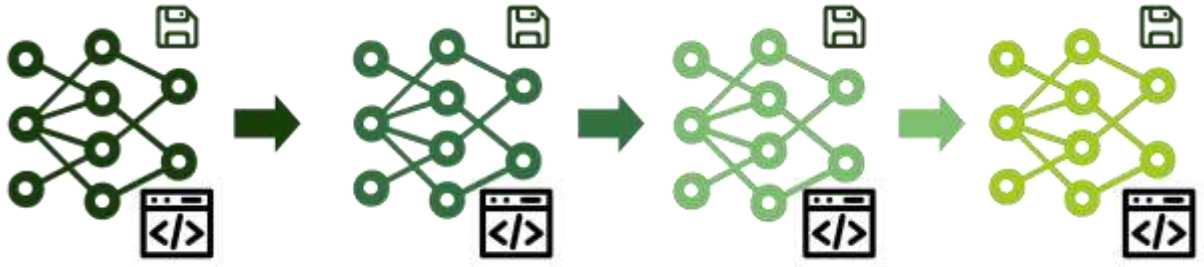


Model – versioning

- Whilst creating and finetuning → different model versions
  - Different hyperparameters (learning rate, batch size …)
  - Different model architectures
  - Different data
  - Different resulting metrics (accuracy, loss …)

  = METADATA

  → Keep track of the different models and their metadata

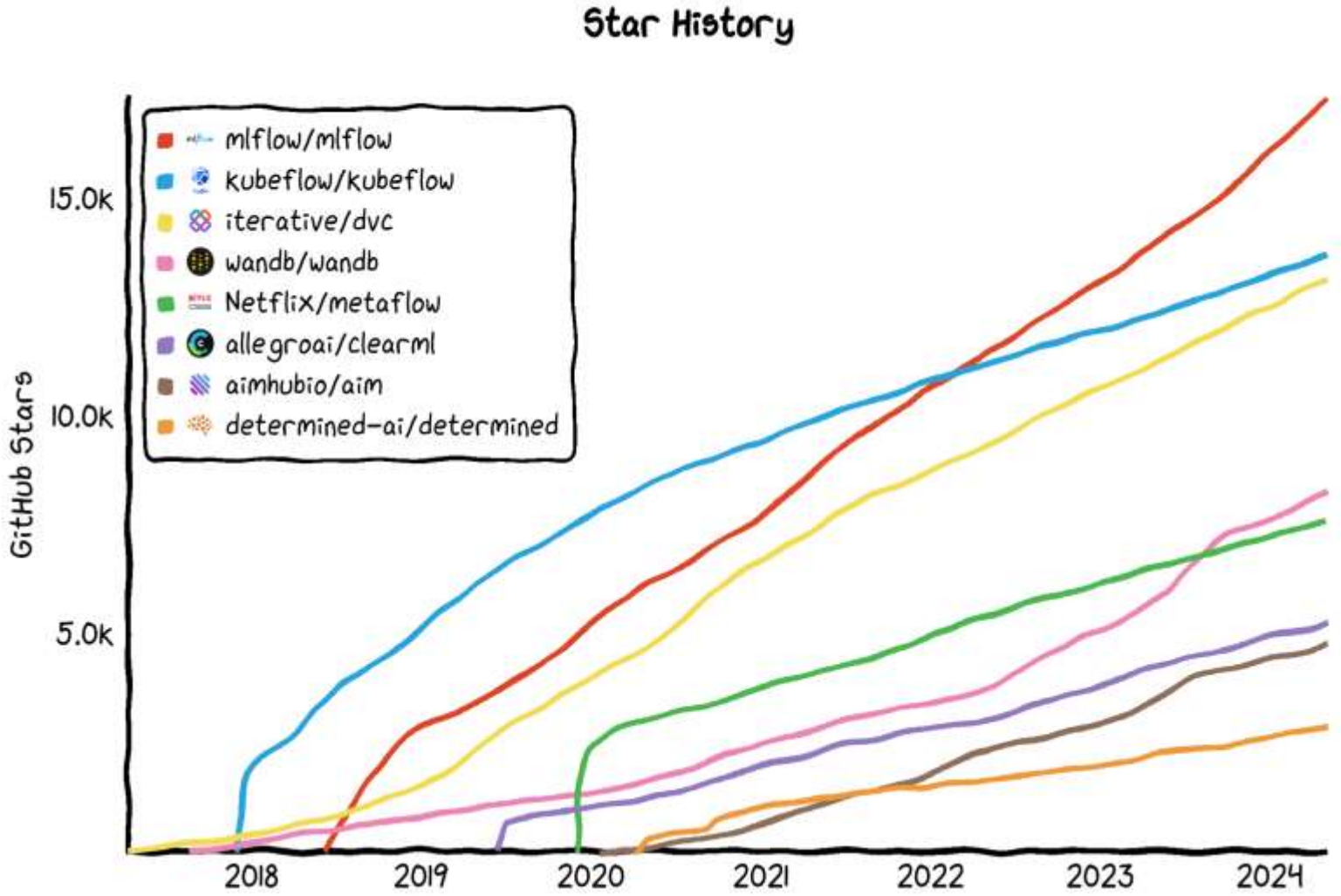


Models – tools

- Experiment tracking
- MLflow, Weights and

Weights & Biases

mlflow



Model – MLflow

- Open-source project to manage ML lifecycle
- Python package
- MLflow model tracking (logging parameters)
- Mlflow projects organize implementation code in reproducible way
- Mlflow model packaging for serving
- Model registry version models and have model lineage

Demo – MLflow

- Python library

```
import mlflow
import mlflow.pytorch
```

MLflow – code

```
# TRAIN MODEL
mlflow.set_tracking_uri(uri="http://127.0.0.1:8080")
mlflow.set_experiment("MLOps-DEMO")
mlflow.pytorch.autolog(disable=True)

with mlflow.start_run(run_name = "mlops-demo"):
    train_params = {
            "learning_rate": learning_rate,
            "weight_decay": weight_decay,
            "epochs": epochs,
            "batch_size": batch_size
    }
    binary_classifier= Binary_classifier()
    model_params = binary_classifier.model_params()
    mlflow.log_params(model_params)
    mlflow.log_params(train_params)
    train_params, train_metrics, val_metrics, model = train(

    mlflow.log_metric("val_best_epoch", val_metrics['best epoch'])
    ref_preds, val_acc, val_loss = eval("models/", binary_classifier, loaded_data.val_data)
    mlflow.pytorch.log_model(model, "mlops demo")
```

MLflow – code

train_utils.py

src > utils > train_utils.py > Train_utils > evaluation_metrics > acc

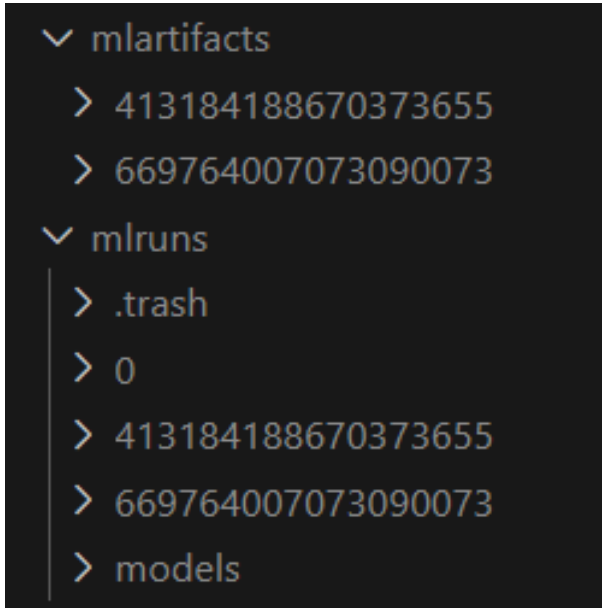
```
18  class Train_utils(object):
116
117     def train(self):
118         best_acc = - np.inf
119         best_loss = np.inf
120         best_epoch = -1
121         best_weights = None
122         num_batches = int(len(self.inputs_train)/self.batch_size)
123
124         all_losses = []
125         all_accuracies = []
126         for epoch in range(self.num_epochs):
127             # TRAIN THE MODEL
128             self.bin_class.train()
129             train_acc, train_loss = 0, 0
```

```
mlflow.log_metric("train_acc", train_acc/num_batches,step=epoch,)
mlflow.log_metric("train_loss", train_loss/num_batches, step=epoch)
train_metrics = {
```

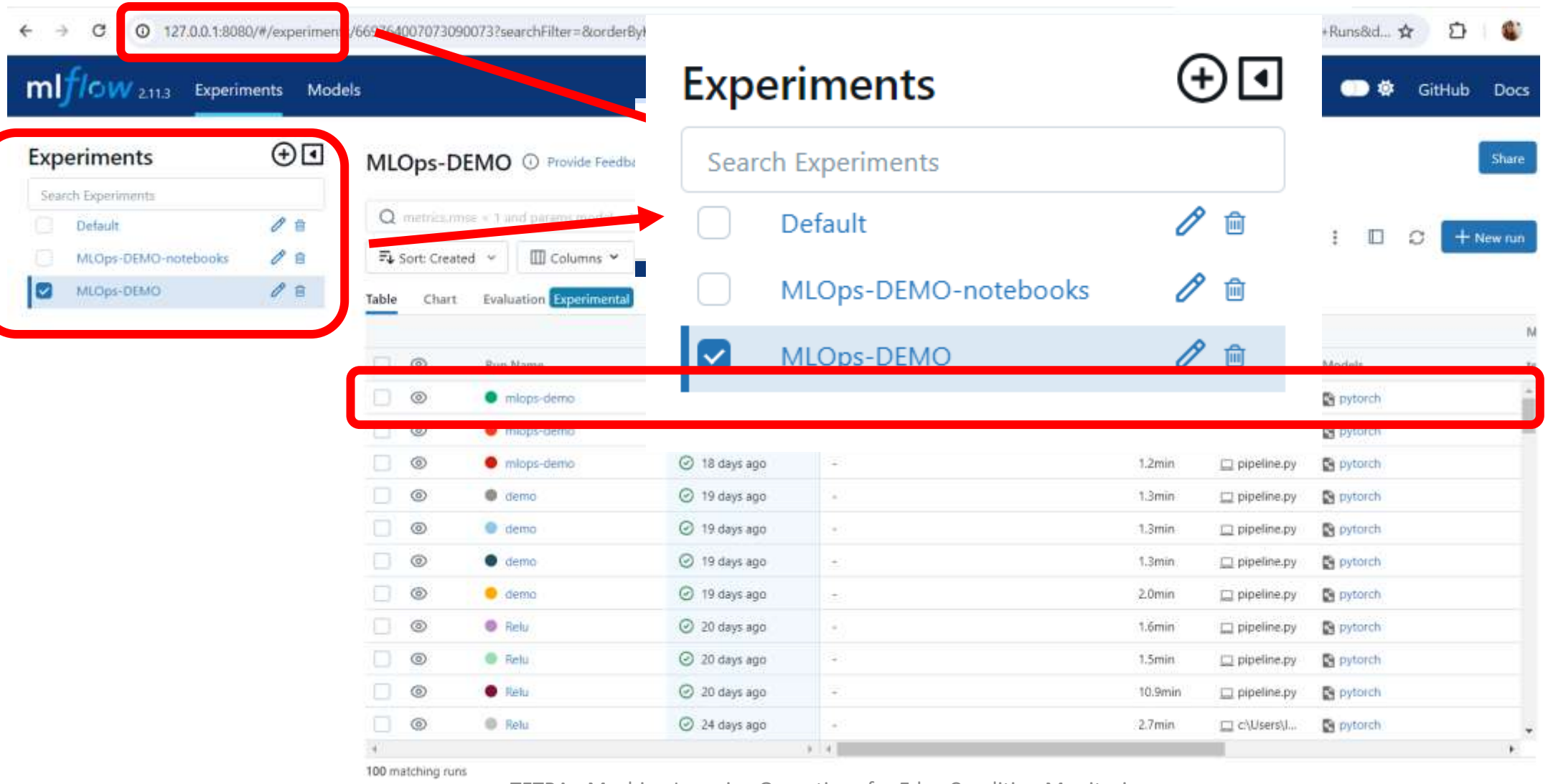
MLflow – server

```
PS C:\Users\laral\Visual_studio_code\mlops-demo> mlflow server --host 127.0.0.1 --port 8080
INFO:waitress:Serving on http://127.0.0.1:8080
```

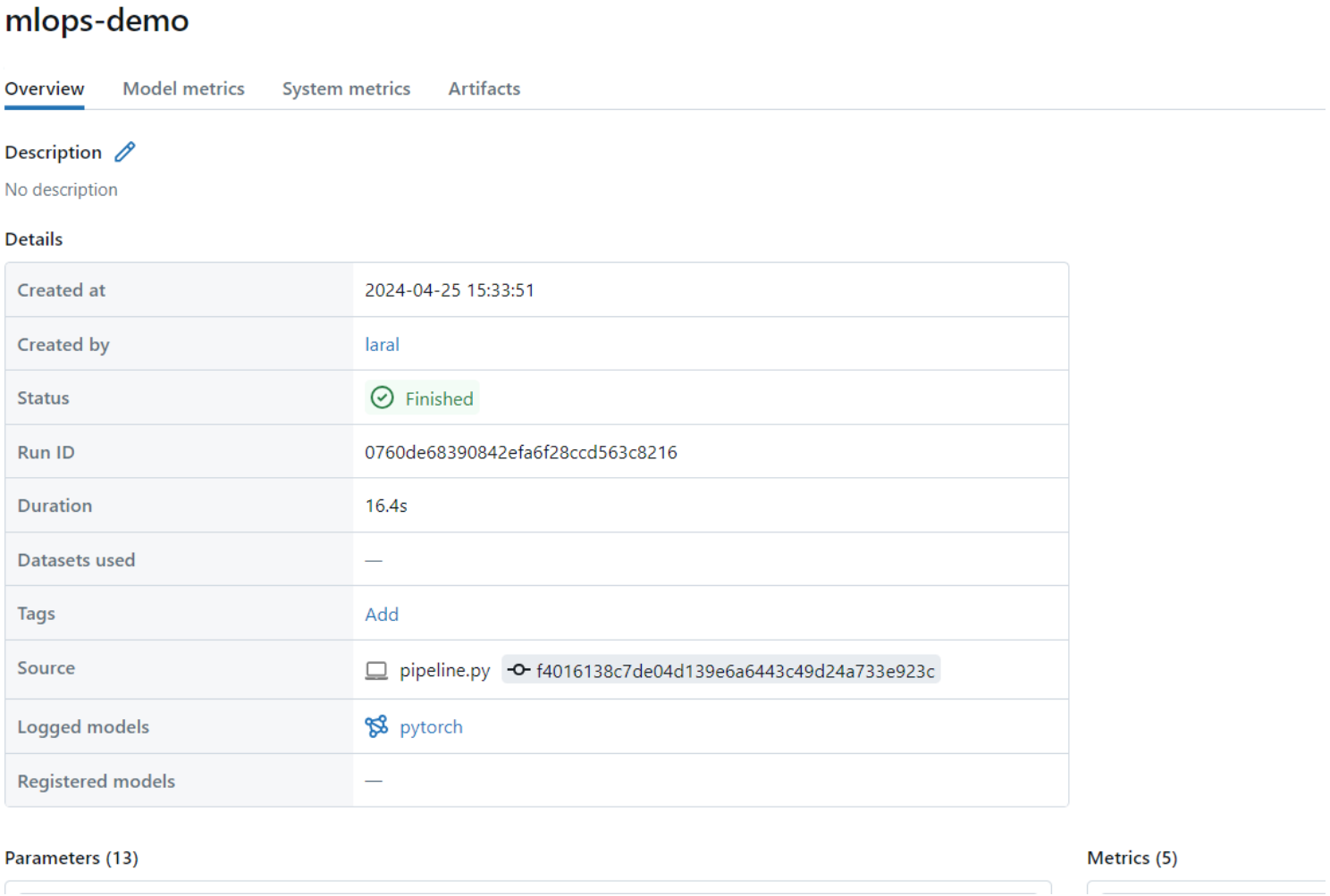
Run the code creates mlartifacts and mlruns



MLflow – server



MLflow – server



MLflow – server

Parameters (13)

Search parameters

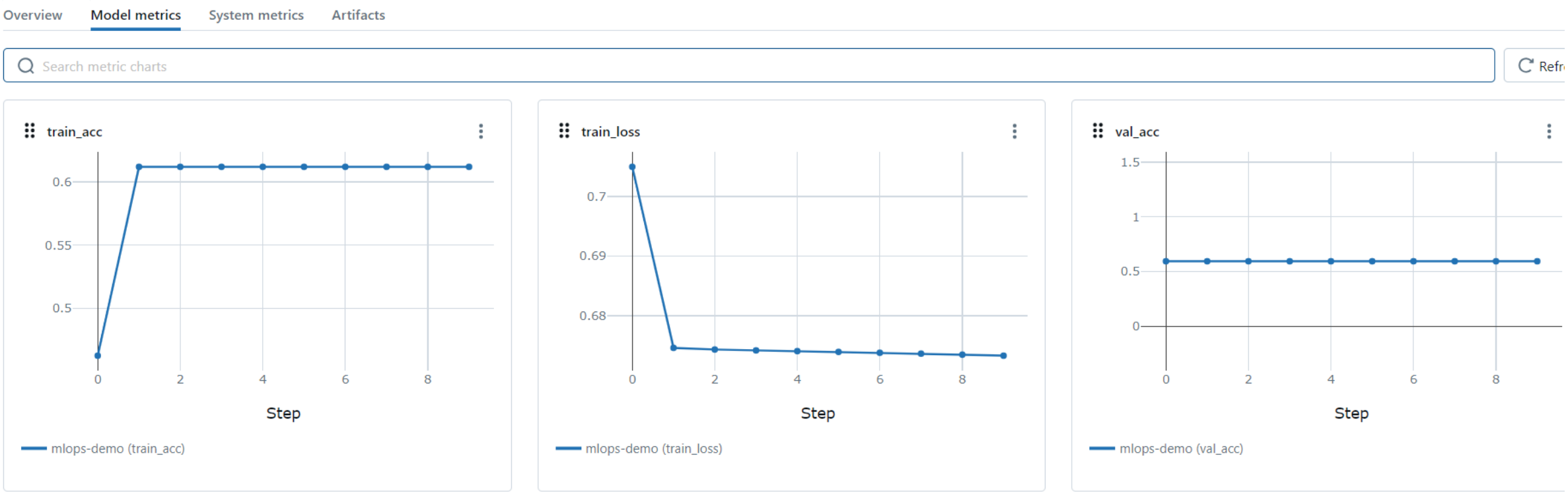
Parameter	Value
activation_out	sigmoid
batch_size	16
epochs	10
learning_rate	0.0001
num_hl1	50
num_hl2	128
num_hl3	512
num_hl4	256
num_hl5	128
num_hl6	64
num_hl7	15

Metrics (5)

Search metrics

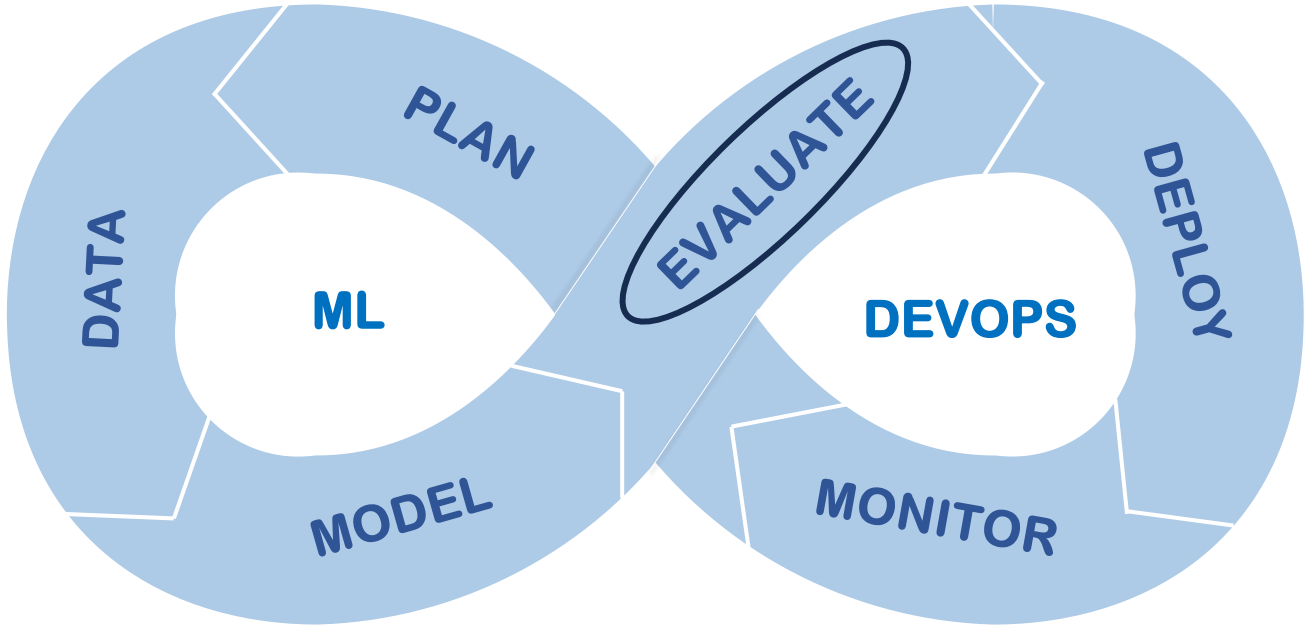
Metric	Value
train_acc	0.6118353605270386
train_loss	0.6732871532440186
val_acc	0.5945122241973877
val_best_epoch	0
val_loss	0.6740019917488098

MLflow – server



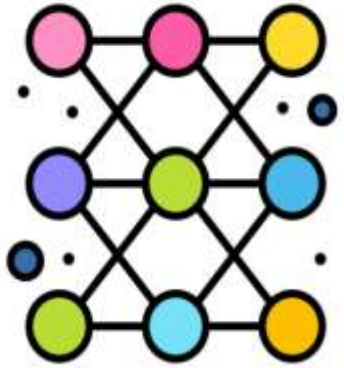
The pipeline

- Plan
- Data
- Model
- **Evaluate**
- Deploy
- Monitor
- Closing the loop



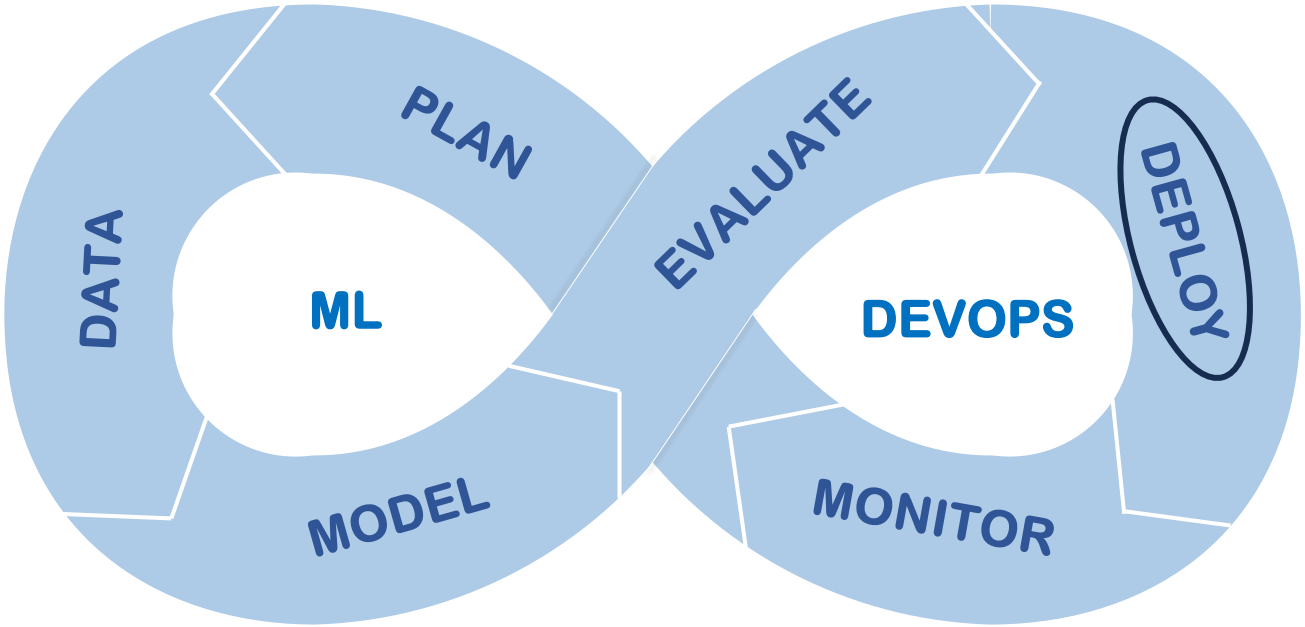
Evaluate – what

- Evaluate the created machine learning model in context
  - Choose metrics: accuracy, resource usage, fairness, business KPIs ...
- If there is already a model deployed
  - Champion/challenger testing
  - A/B testing



The pipeline

- Plan
- Data
- Model
- Evaluate
- **Deploy**
- Monitor
- Closing the loop



Deploy – what

- Prepare for production:
  - Model compression (vb. Quantization, pruning, model distillation)
  - Change to correct format (vb. ONNX)
  - Risk mitigation
- Create CI/CD pipelines
- Containerization = Docker

Deploy – tools

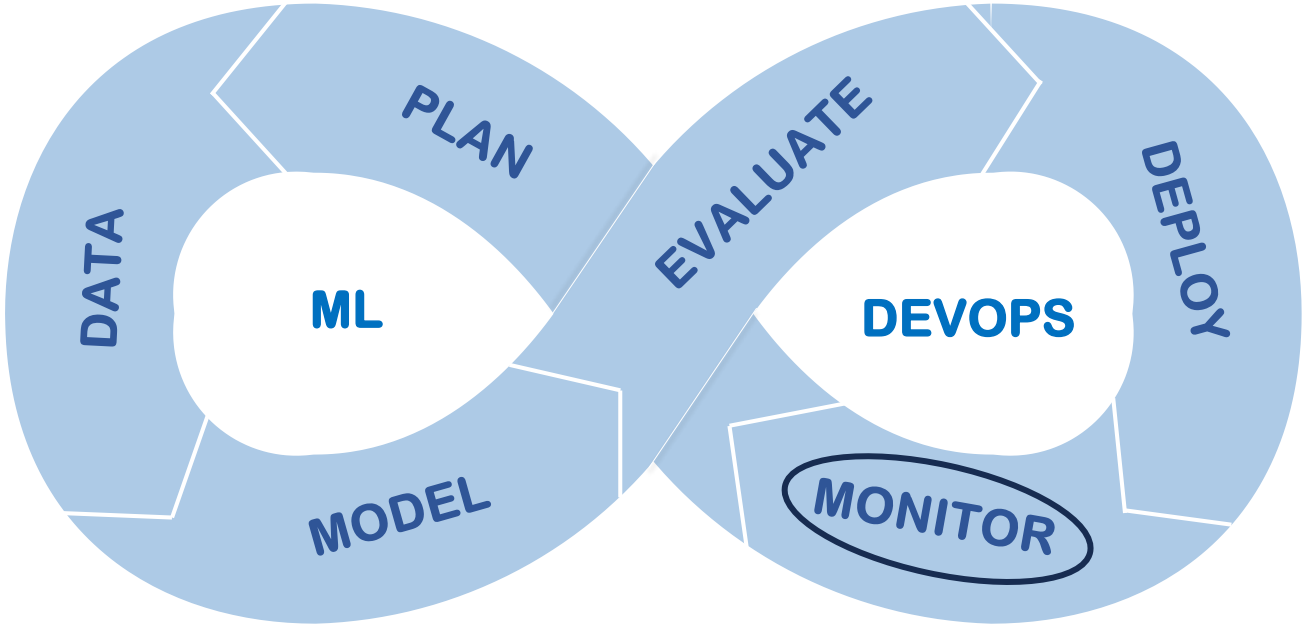
- Containerization: Docker, Kubernetes
- Deployment tools: Seldon.io, TensorFlow serving, Kubeflow, AWS Sagemaker, BentoML …

Amazon SageMaker

Kubeflow

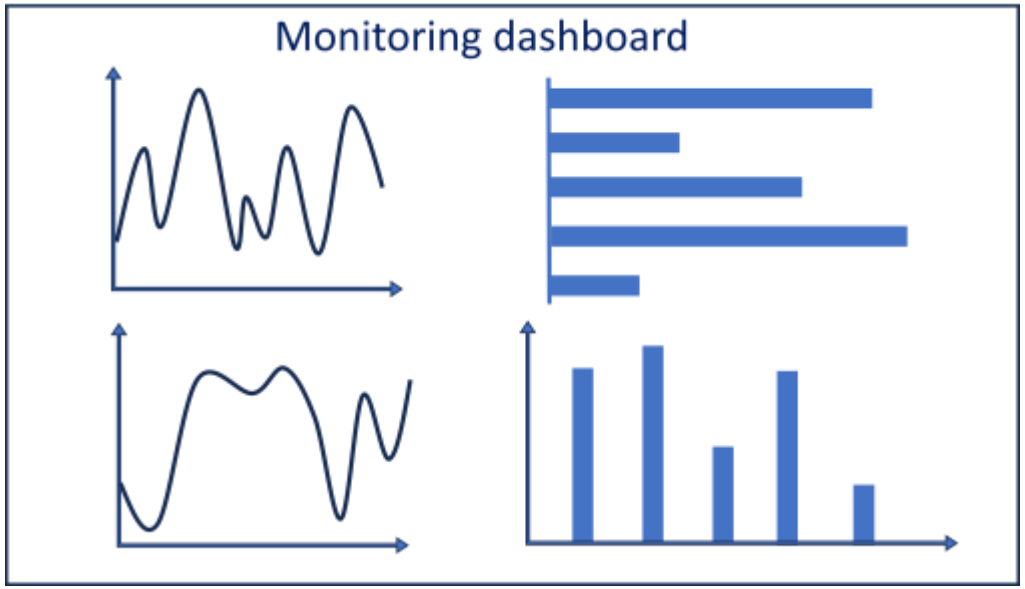
The pipeline

- Plan
- Data
- Model
- Evaluate
- Deploy
- **Monitor**
- Closing the loop



Monitor – what

- Monitor the data for data quality and data drift
- Monitor the machine learning model for concept drift
- Monitor other parameters for drift e.g. resources, fairness …



Monitoring – tools

- Tools that monitor input data, predictions and model quality
- Evidently, Arize, Whylabs ...

Monitoring – Evidently

- Open-source ML observability platform
- Python library → create (pre-made) test suits and reports
- Able to save as html, json, dictionary → fully offline possible
- Only tabular and text data → working on other unstructured data

Monitoring – Data quality

- Use reference dataset
  - Expected schema and column types
  - Expected batch size
  - Statistics: averages, min-max ...
  - …
- Use thresholds
  - Share of missing values
  - Duplicate columns/rows
  - Constant features
  - ...

Monitoring – Demo

Demo: Evidently – Model quality

Evidently – python library

```python
from evidently import ColumnMapping
from evidently.report import Report
from evidently.tests import *
from evidently.metrics import *
from evidently.test_suite import TestSuite

# data drift
from evidently.calculations.stattests import StatTest
from evidently.metric_preset import DataDriftPreset

# data quality
from evidently.metric_preset import DataQualityPreset
from evidently.test_preset import DataQualityTestPreset

# Model quality
from evidently.metric_preset import ClassificationPreset

# prediction drift
from evidently.metric_preset import TargetDriftPreset
```

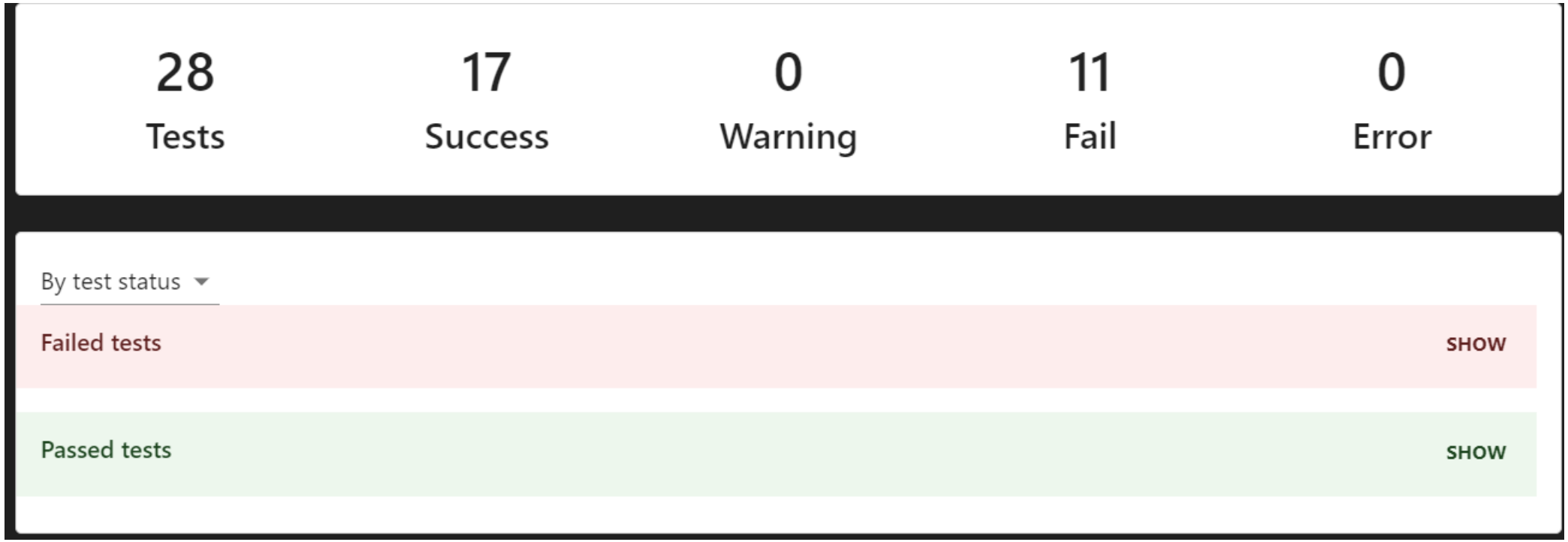
Data quality – test suite

```
data_quality = TestSuite(tests=[
    DataQualityTestPreset(),
])

data_quality.run(reference_data=df_reference, current_data=df_test, column_mapping=cm)
data_quality.show(mode="inline")
```

```
data_quality_column_tests = TestSuite(tests=[
    TestColumnValueMean(column_name='ph'),
    TestValueRange(column_name='Conductivity'),
    TestShareOfOutRangeValues(column_name='Conductivity'),
    TestNumberOfOutListValues(column_name='Hardness'),
    TestColumnQuantile(column_name='Organic_carbon', quantile=0.25),
    TestColumnShareOfMissingValues(column_name='Trihalomethanes'),
])
```

```
data_quality.run(reference_data=df_reference, current_data=df_test, column_mapping=cm)
data_quality.show(mode="inline")
data_quality_column_report.save_html( save_dir + str("data_quality_column_report.html"))
```

Data quality – test suite



Data quality – test suite

Share of the Most Common Value — DETAILS

The most common value in the column **Potable** is 0.4435722529888153. Its share is 0.003. The test threshold is eq=0.0008 ± 8e-05.

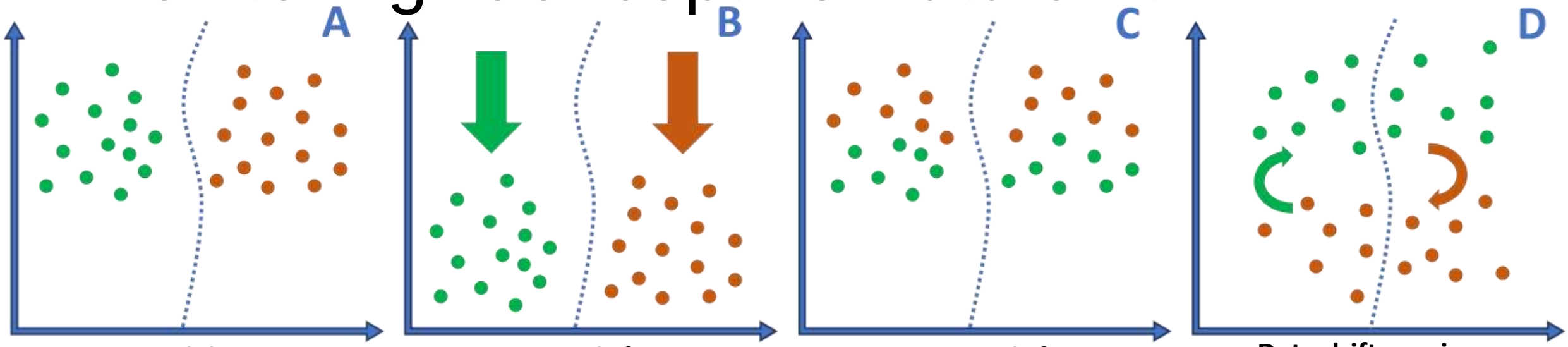
CURRENT VALUE COUNTS (TOP 10) | REFERENCE VALUE COUNTS (TOP 10)

value	count
0.4435722529888153	1.0
0.30121520161628723	1.0
0.3226304054260254	1.0
0.5187572836875916	1.0
0.5885509252548218	1.0
0.7642117738723755	1.0
0.3245012164115906	1.0
0.3078989088535309	1.0
0.3575007915496826	1.0
0.3965248167514801	1.0

Data quality – report

```
data_quality_report = Report(metrics=[
    DataQualityPreset(),
])
data_quality_report.run(reference_data=df_reference,current_data=df_current, column_mapping=cm)
data_quality_report.show(mode='inline')
```

Monitoring: Concept vs Data drift



Original data

Data drift

Data distribution changes but the model prediction is still correct

e.g. classification dog vs. Cat gets cats in different colors

Concept drift

Data distribution stays the same, but the model prediction is incorrect

e.g. prediction housing price gets more expensive over time

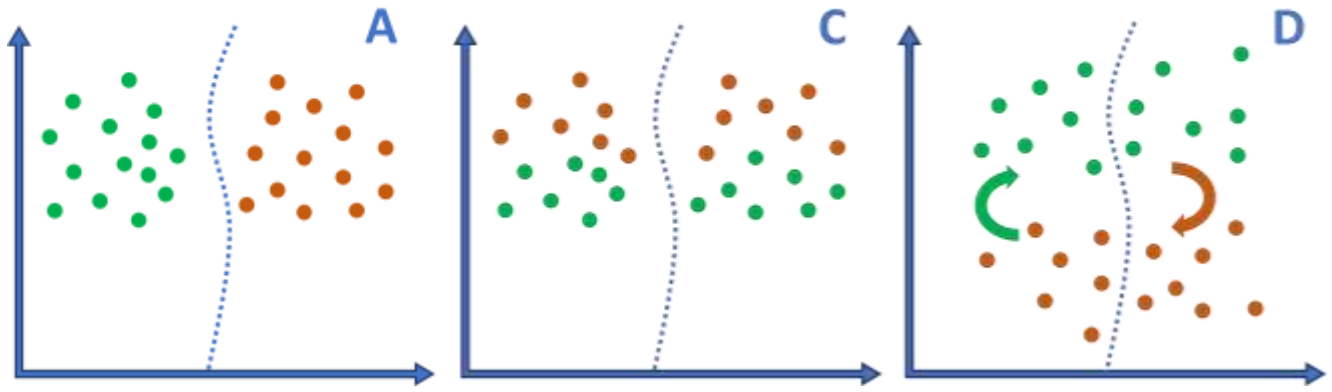
Data drift causing concept drift

Data distribution changes and the model prediction is incorrect

e.g. different machine settings lead to wrong predictions

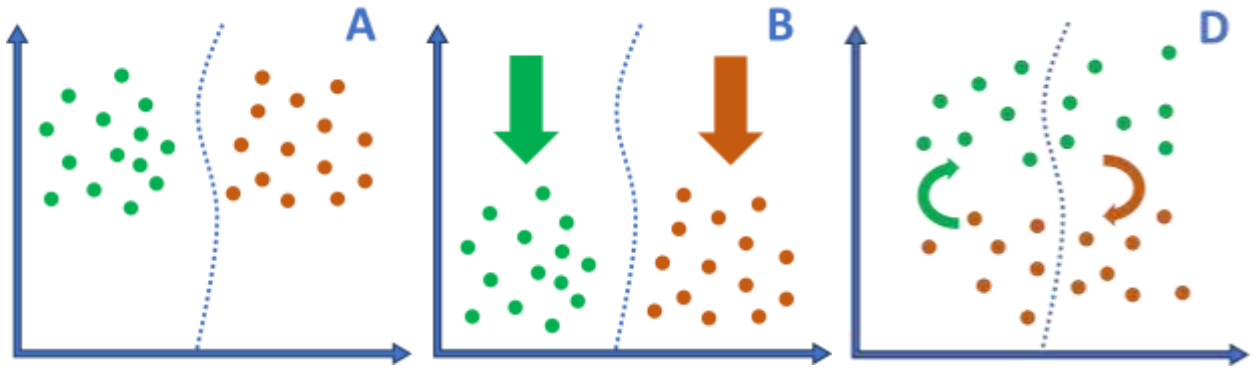
Monitoring – Concept drift detection

- The distribution of the output target changes
- If true labels = known → monitor model quality
- Otherwise → prediction distribution monitoring
  - Statistical tests
  - Distribution-based metrics
  - Machine learning model



Monitoring – Data drift detection

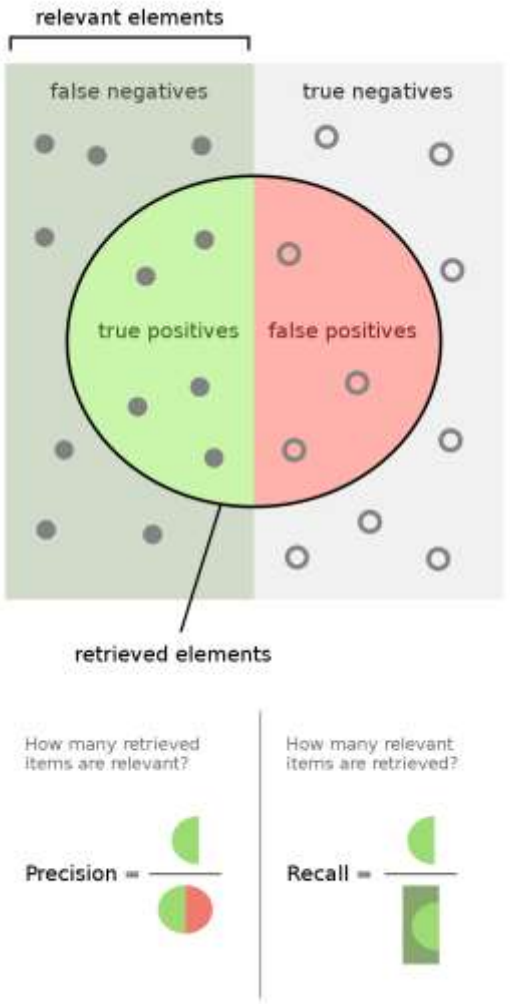
- Input data distribution monitoring
  - Statistical tests
  - Distribution metrics
  - Machine learning model

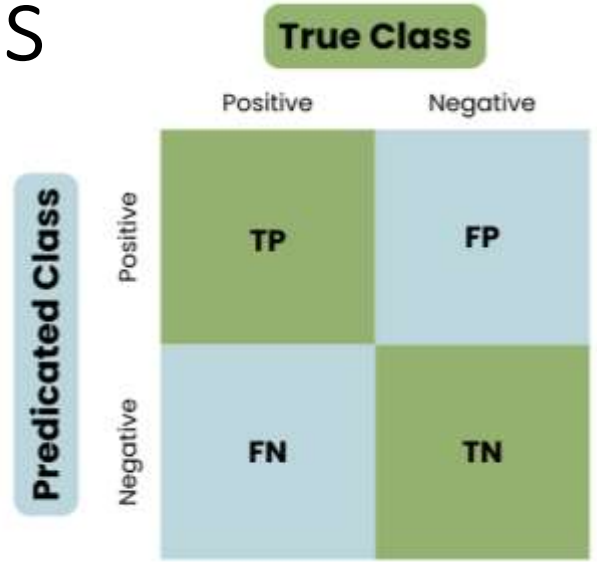


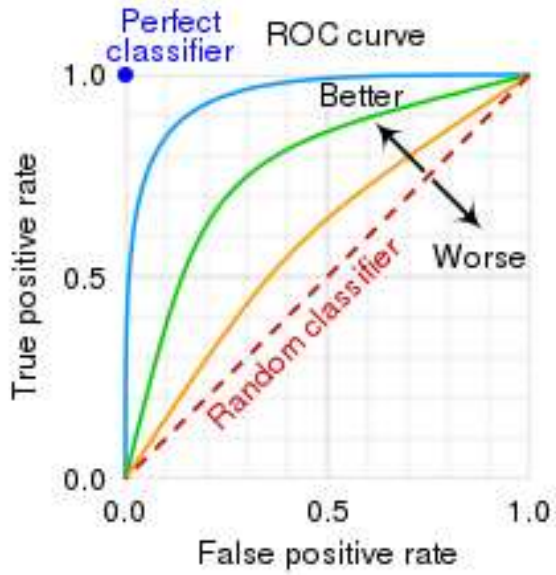
- If no true labels → monitor predictions over input data

Monitoring – Model quality metrics

- Accuracy
- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- Confusion matrix
- Recall
- Precision
- ROC
- …







Monitoring – Demo

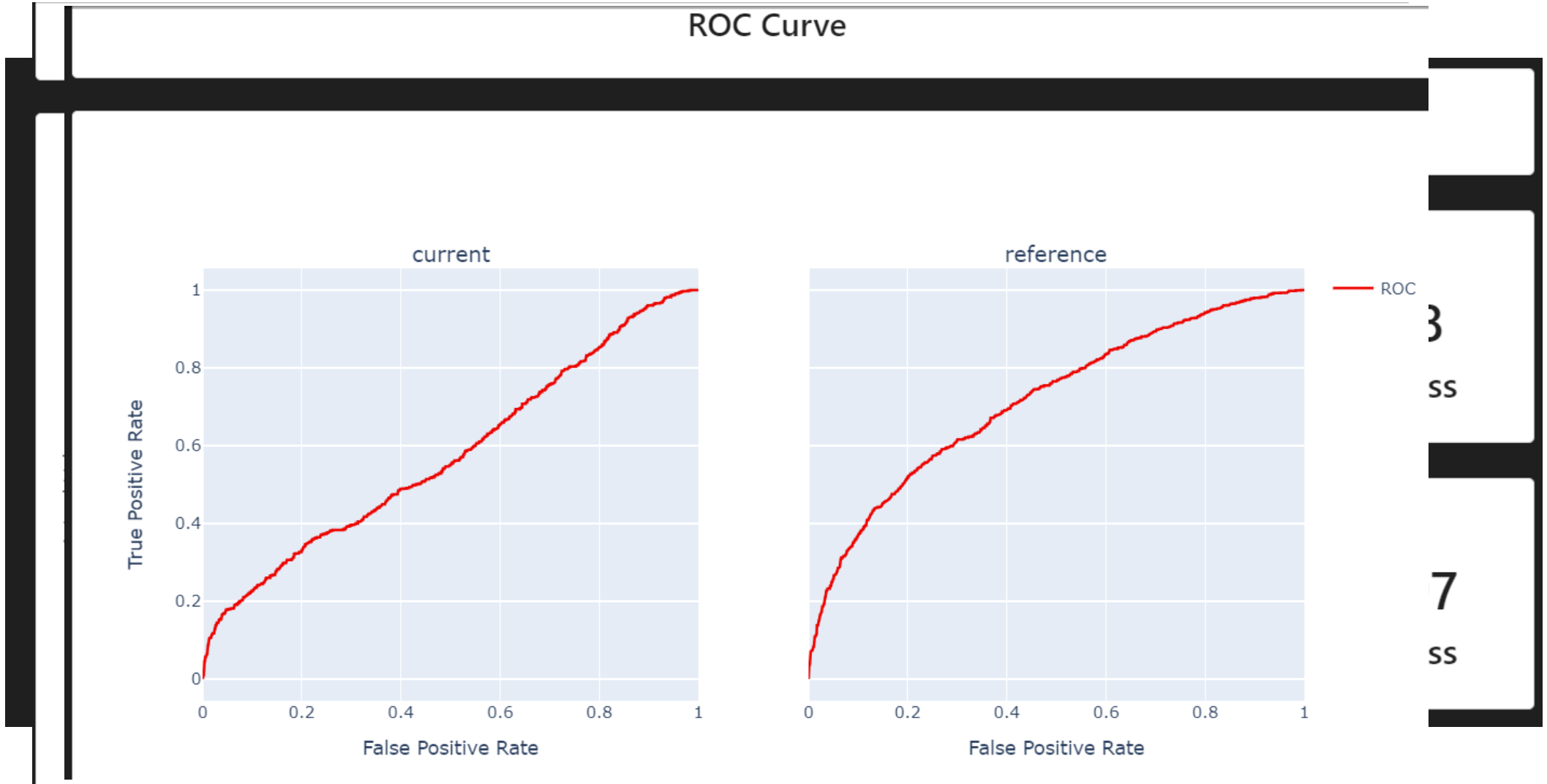
Demo: Evidently – Model quality

Evidently – model quality metrics

```
classification_performance_report = Report(metrics=[
    ClassificationPreset(),
])

classification_performance_report.run(reference_data=df_reference, current_data=df_test, column_mapping=cm)
classification_performance_report.show(mode='inline')
```

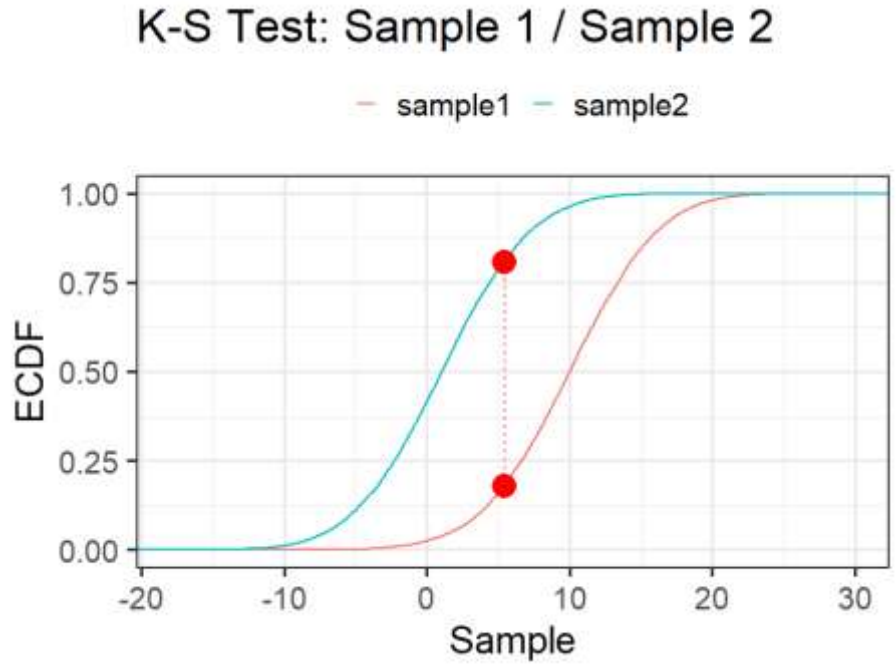
```
classification_report = Report(metrics=[
    ClassificationQualityMetric(),
    ClassificationClassBalance(),
    ConflictTargetMetric(),
    ConflictPredictionMetric(),
    ClassificationConfusionMatrix(),
    ClassificationRocCurve(),
    ClassificationPRCurve(),
    ClassificationPRTable(),
])
classification_report.run(reference_data=df_reference, current_data=df_current, column_mapping=cm)
classification_report.show(mode='inline')
```

Evidently – model quality metrics



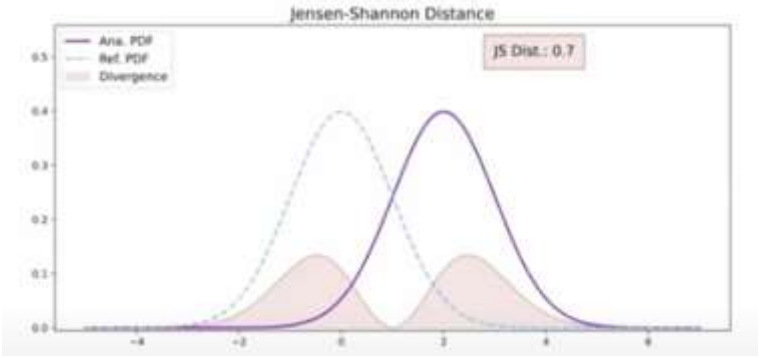
Monitoring – Statistical tests

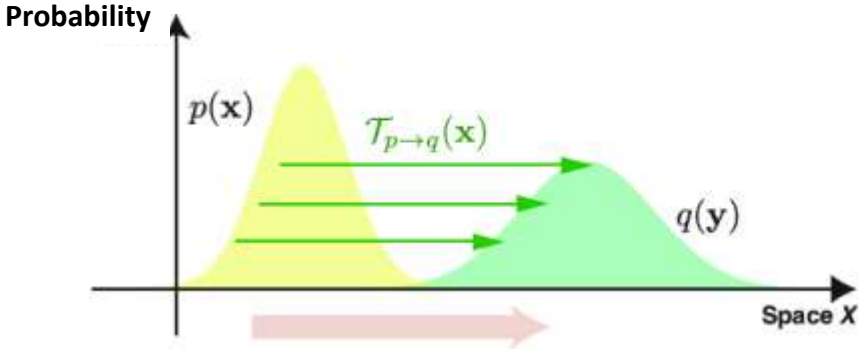
- Compare two statistical distributions (reference and current):
  - Kolmogorov-Smirnov
  - Chi-squared
  - …
- More sensitive in smaller distributions



Monitoring – Distribution distance metrics

- Comparison of two distributions
  - Jensen-Shannon distance
  - Wasserstein distance
  - …
- Better for larger distributions



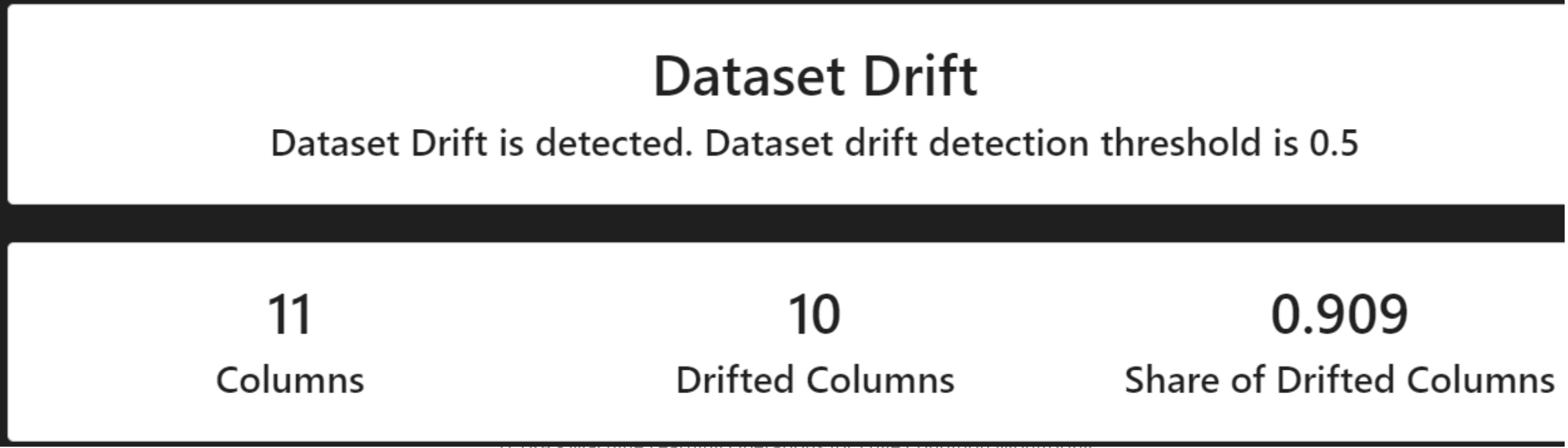


Monitoring – Demo

Demo: Evidently – Model quality

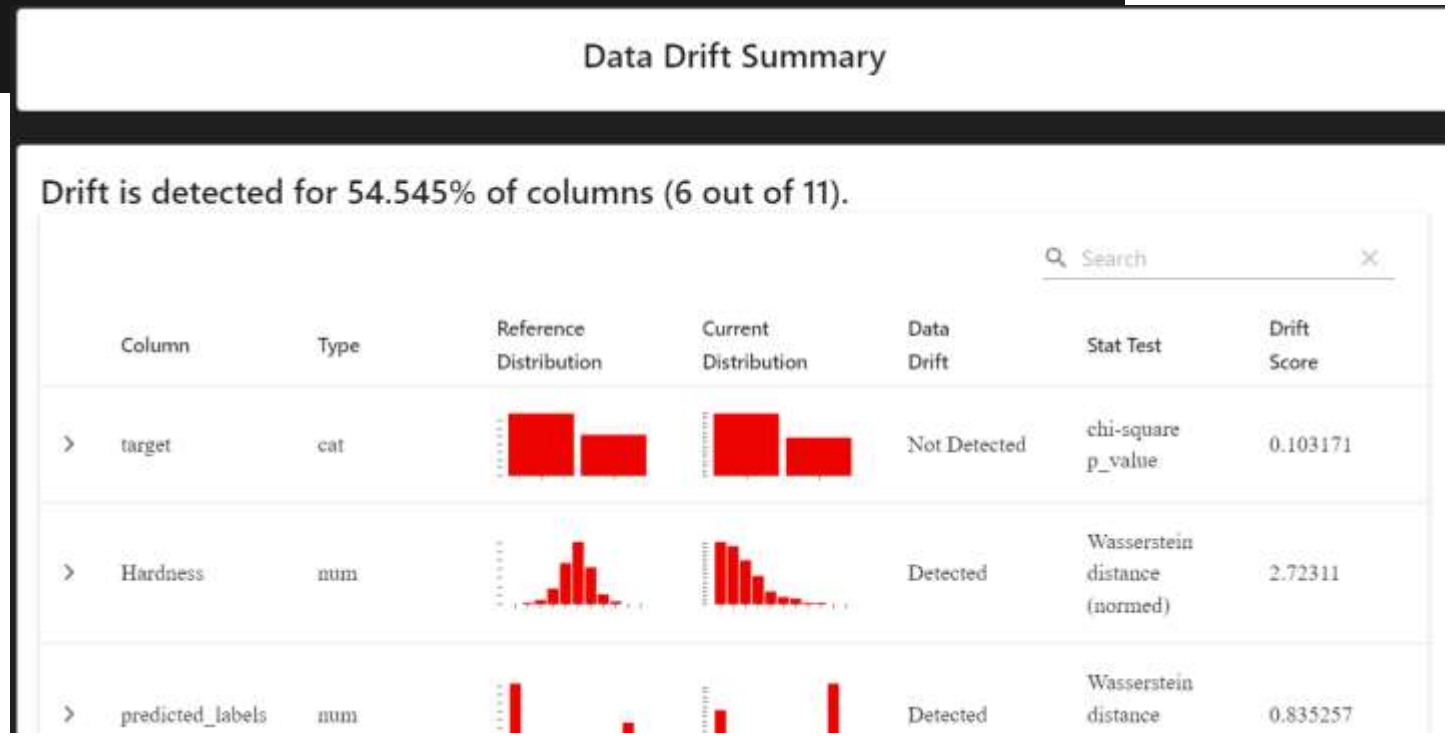
Evidently – drift detection

```python
# DRIFT REPORT share of drifted features whole dataset
data_drift_share_report = Report(metrics=[
    DatasetDriftMetric()
])
data_drift_share_report.run(reference_data=df_reference, current_data=df_current, column_mapping=cm)
data_drift_share_report.show(mode='inline')
```



Evidently – drift detection

```
# DRIFT REPORT FULL DATASET BUT OTHER TEST DEPENDING NUM OR CAT
data_drift_dataset_report = Report(metrics=[
    DataDriftTable(num_stattest='wasserstein', cat_stattest='chisquare'),
])
data_drift_dataset_report.run(reference_data=df_reference, current_data=df_current, column_mapping=cm)
data_drift_dataset_report.show(mode='inline')
```



Data Drift Summary

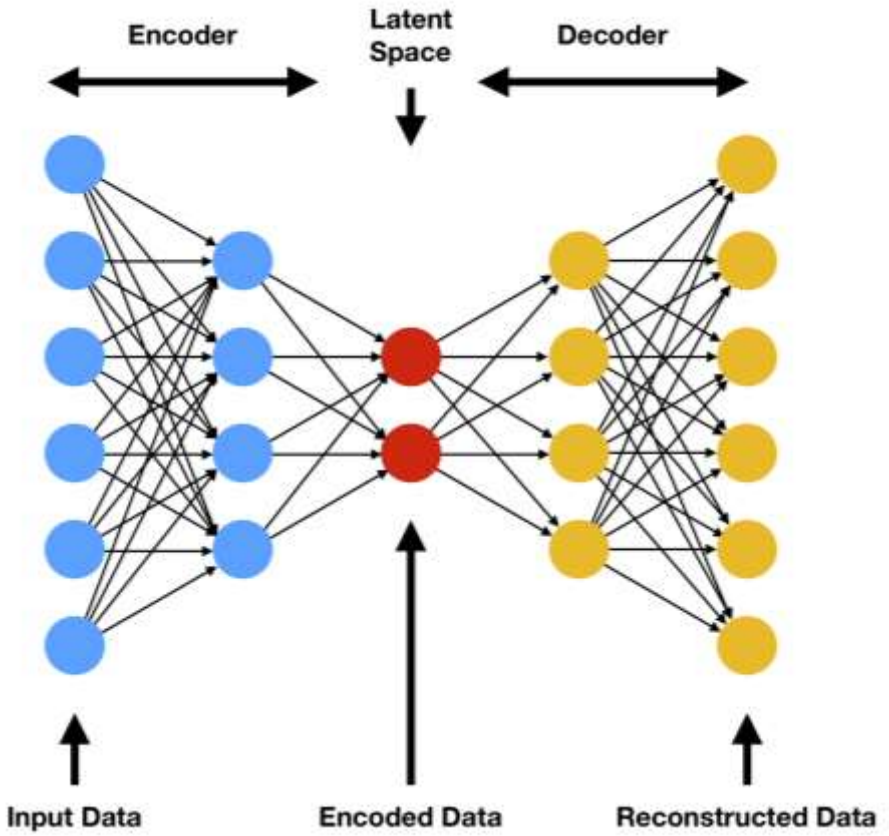
Drift is detected for 54.545% of columns (6 out of 11).

Search

	Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
>	target	cat			Not Detected	chi-square p_value	0.103171
>	Hardness	num			Detected	Wasserstein distance (normed)	2.72311
>	predicted_labels	num			Detected	Wasserstein distance	0.835257

Monitoring – Machine learning model

- Deep neural network
- K nearest neighbors
- autoencoder
- …

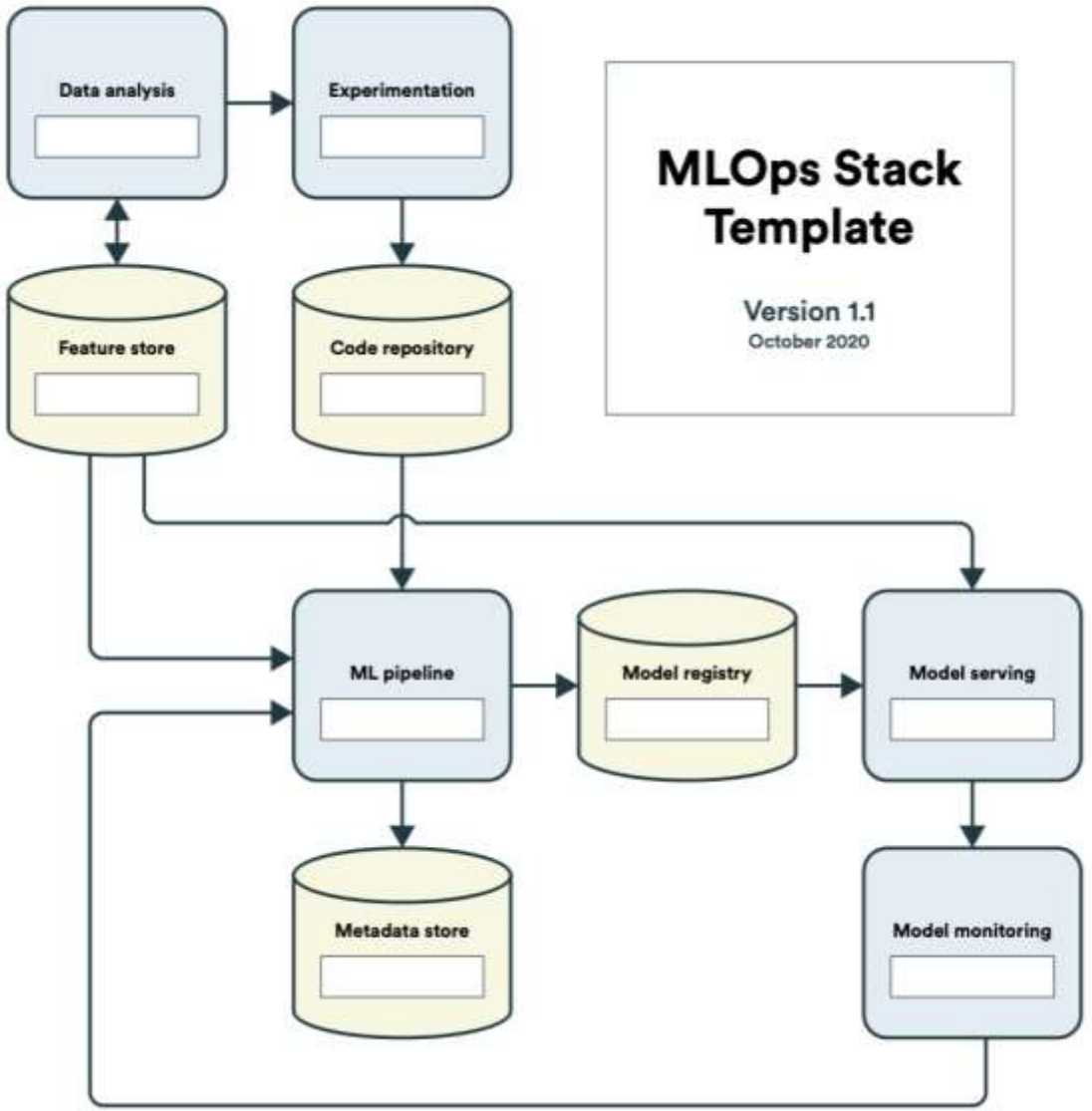


Monitoring – other types of drift

- Business metrics / Key Performance Indicators
- Fairness
- Resource consumption
  - Memory usage
  - Processing power
- …

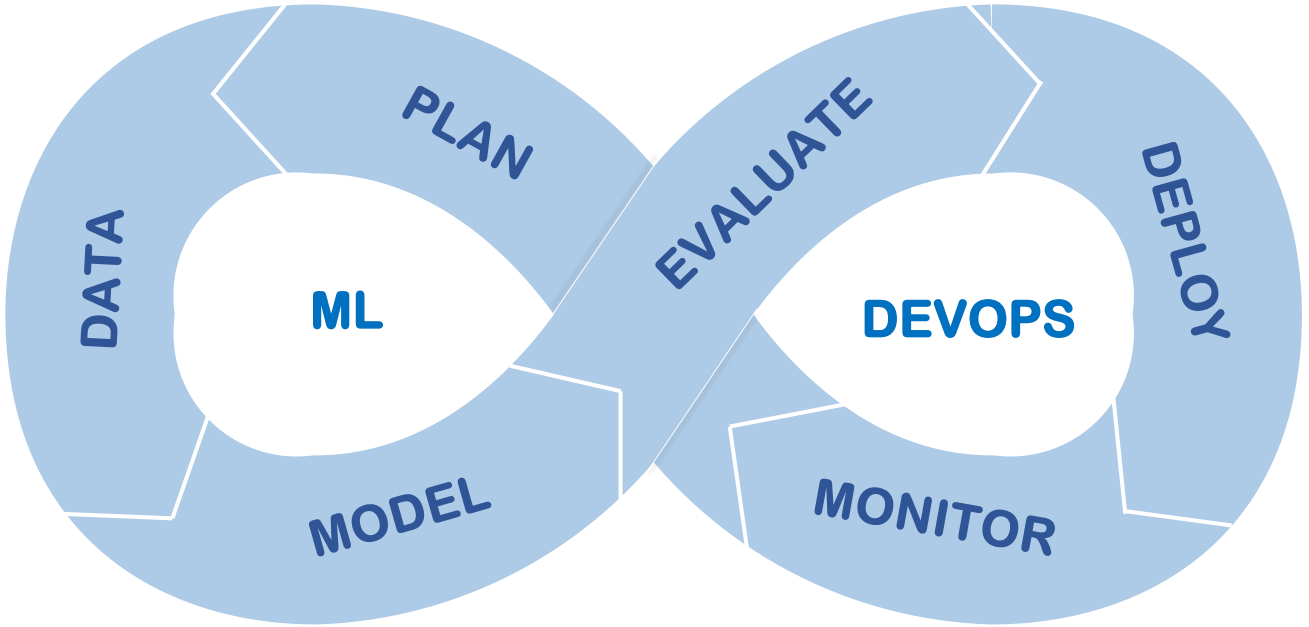
MLOps stack

- Scope of each tool might span several components of the MLOps process
  - ➔Requires careful consideration depending in use case requirements
- MLOps stack template as guideline
- As for regular software development: thorough requirements analysis upfront!



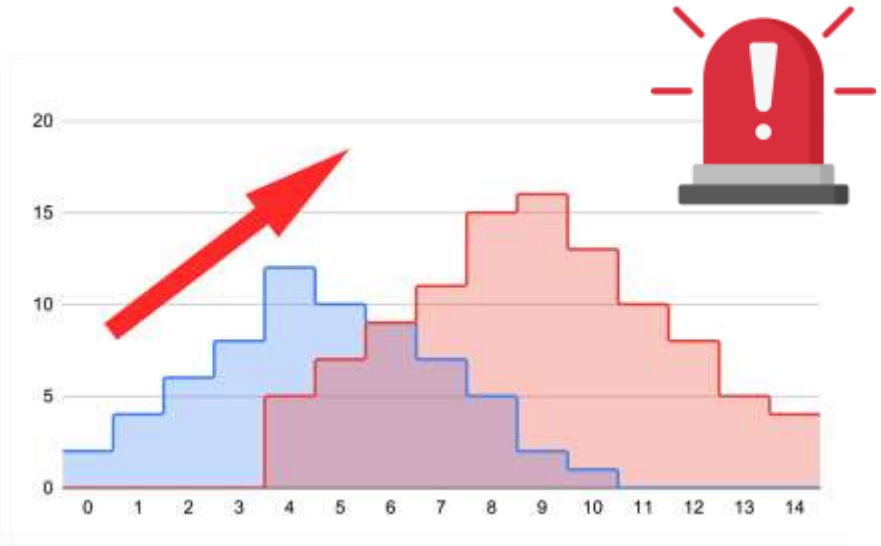
The pipeline

- Plan
- Data
- Model
- Evaluate
- Deploy
- Monitor
- **Closing the loop**



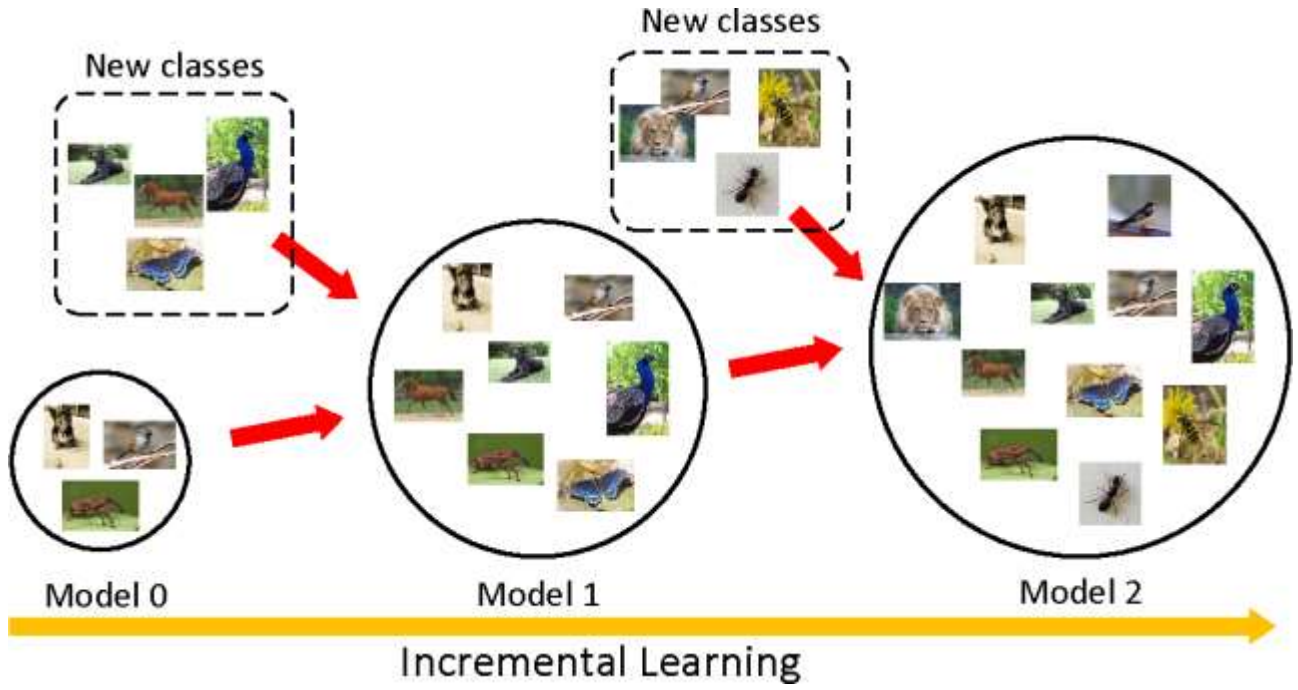
Closing the loop – when?

- When to adapt your model
  - Scheduled: daily, weekly, monthly...
  - When drift is detected



Closing the loop - How

- Types of model adaptation techniques
  - Creating a new model from scratch
  - Lazy learning
  - Incremental learning
  - Ensemble method



ONNX & Edge Deployment

Alexander D’hoore

ONNX & Edge Deployment

1. Introduction: ONNX
2. General tools for ONNX
3. Platform specific tools
4. TETRA hardware proposal
5. Conclusion

Introduction: ONNX

Introduction: ONNX

- ONNX: Open Neural Network Exchange
- “The open standard for machine learning interoperability”

- Announced by Facebook and Microsoft in 2017
- Supported by a LARGE list of companies

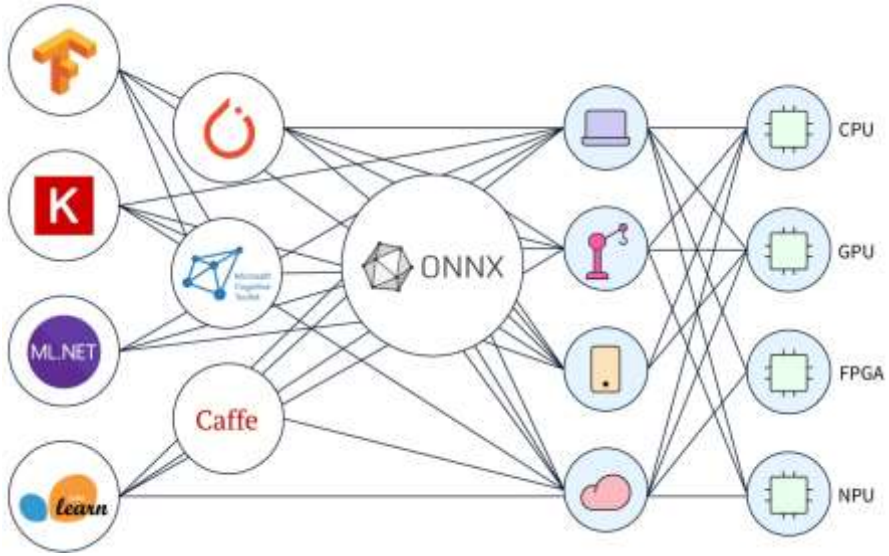
ONNX

- Open file format that describes AI models,
- both **deep learning and traditional ML**

ONNX: Partners

ONNX: Key benefits

- **Framework interoperability**
  - Train model in your preferred framework
  - Export model to ONNX for production
  - Teams can choose their own tools
- **Shared optimization**
  - Easy to optimize multiple frameworks at once
  - Hardware specific optimizations by vendors

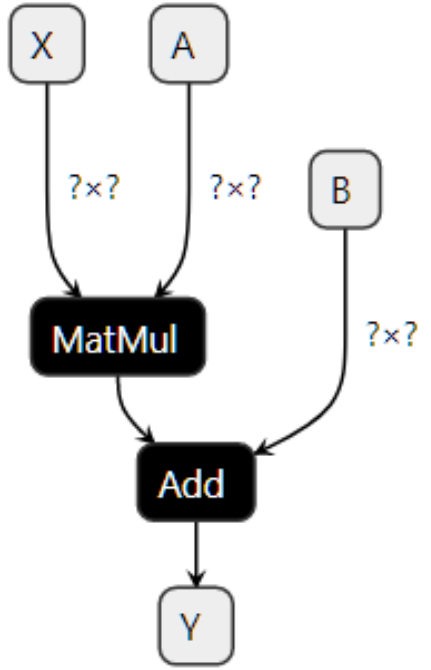


ONNX: File format

- Provides an extensible **computation graph** model
- Defines built-in **operators** and standard **data types**

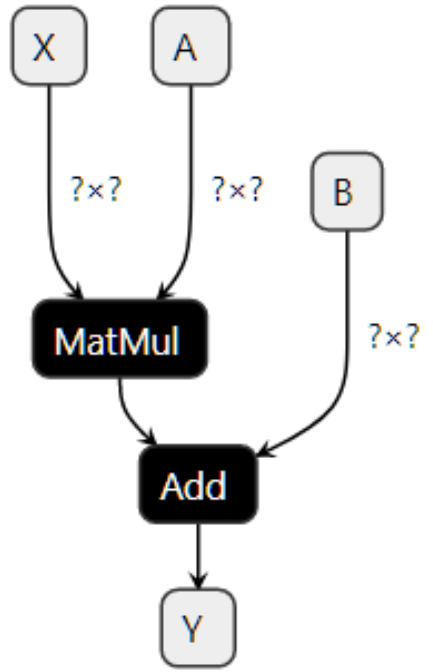
- Focuses on inferencing (evaluation)
  - Support for training graphs in “preview”

- **Computation Graph**
  - A list of nodes forming an acyclic graph
  - Nodes have one or more inputs and outputs
  - Each node represents a call to an operator



ONNX: File format

- **Built-in Operators**
  - Implemented by ONNX-supporting frameworks
  - Neural networks: MatMul, Conv, DropOut, Relu…
  - Traditional ML: TreeEnsemble, SVMRegressor…
- **Standard data types**
  - Numerical computation with tensors
  - Basic: float, double, int8, int16, bool…
  - Advanced: IEEE float16, Google bfloat16



General tools for ONNX

General tools for ONNX

- We will go over a few tools that support ONNX
  - There are MANY more, we can’t cover all of them

- We are interested in:
  - Tools for **training**
  - Tools for **optimization**
  - Tools for **inference**

- These categories are not strictly separated
  - Tools often have overlapping functionality

ONNX Runtime

- High-performance **inference engine** by Microsoft
  - Often confused with ONNX standard (not the same)
  - Recently added support for on-device training
- Open-source and cross-platform
  - Good option on Linux, no bare metal
- Supports hardware acceleration
  - Intel CPU/GPU, NVIDIA CUDA GPU,
  - Android NNAPI, Apple Core ML, etc…

PyTorch

- **Deep learning** framework by Facebook
  - Very popular in academia and industry
- Export models to ONNX is officially supported
  - Import ONNX is possible with third party tools
- Support for inference on edge:
  - TorchScript works, but has large dependencies
  - PyTorch Edge (ExecuTorch) is in beta
  - Model quantization is in beta

TensorFlow

- **Deep learning** framework by Google
  - Previously dominant, still widely used
- Export using official tool from ONNX project
  - Import ONNX is deprecated, future unclear
- Support for inference on edge:
  - Offers robust quantization techniques
  - TensorFlow Lite is very mature runtime
  - Supports bare metal microcontrollers

PyTorch vs TensorFlow

- PyTorch is most popular library for training models
  - But lacks lightweight, efficient inference
- TensorFlow has good support for edge deployments
  - But doesn’t have good tools to import ONNX
- So either:
  - Use TensorFlow for everything (train, optimize, inference)
  - Export to **platform specific framework** for inference

Platform specific tools

NVIDIA TensorRT

- High-performance **inference library** by NVIDIA
  - Targets CUDA-enabled NVIDIA GPUs
- Imports models in ONNX format
  - C++ library for edge deployment
  - Embedded GPUs: **NVIDIA Jetson**
- MANY advanced optimizations
  - Quantization: INT8, INT4 (weight only)
  - Layer and tensor fusion

PLC platforms

- Some PLC vendors support ONNX
  - Develop models using standard MLOps tools

- **Beckhoff TwinCAT 3**
  - Neural Network Inference Engine
  - Machine Learning Server (coming soon)

- **Siemens Industrial AI**
  - AI Inference Server
  - AI Model Manager

Arm processors

- **Arm NN SDK**
  - CPU specific optimizations (SIMD)
  - Cortex-A CPUs, Mali GPUs and Ethos NPUs
  - Supports Linux and Android, no bare metal
  - Can import ONNX, but prefers TensorFlow
- **Arm CMSIS-NN**
  - Cortex-M microcontrollers
  - Supports bare metal and RTOS
  - Integrated in TensorFlow Lite Micro

STM32Cube.AI

- Tool to optimize and deploy neural networks
  - For all **STM32 microcontrollers** (Cortex-M)
  - Built on top of ARM CMSIS-NN
- Loads models in ONNX format
  - Analyse RAM / flash requirements
  - Validate model on any computer
  - Control memory usage by layer
  - Generate C-code for inference

TETRA hardware proposal

TETRA hardware proposal

- Which hardware to buy / evaluate?
  - For use cases in TETRA project
  - Might be useful for workshops

- Need wide range of hardware,
- to test multiple ML frameworks

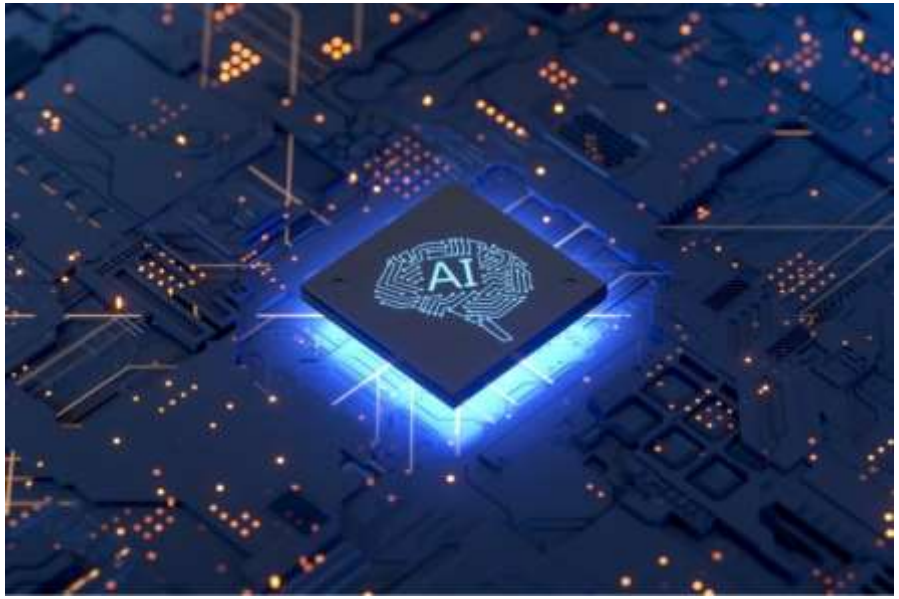
- Feedback is welcome!
  - Talk over lunch

TETRA hardware proposal

- NVIDIA GPU: Jetson Nano
  - TensorRT, ONNX runtime
- Beckhoff / Siemens PLC
- ARM Cortex-A: Raspberry Pi 4/5
  - ARM NN SDK, ONNX runtime, TensorFlow Lite
- ARM Cortex-M: Any STM32 Nucleo
  - STM32Cube.AI, TensorFlow Lite Micro

Conclusion

- **ONNX is here to stay**
  - Most frameworks support ONNX
  - Clear winner for interoperability
- Great future for **embedded AI**
  - Lots of new hardware coming out
  - Optimizations are improving fast
- Look forward to:
  - PyTorch Edge (ExecuTorch)
  - FPGA accelerators (Intel/AMD)



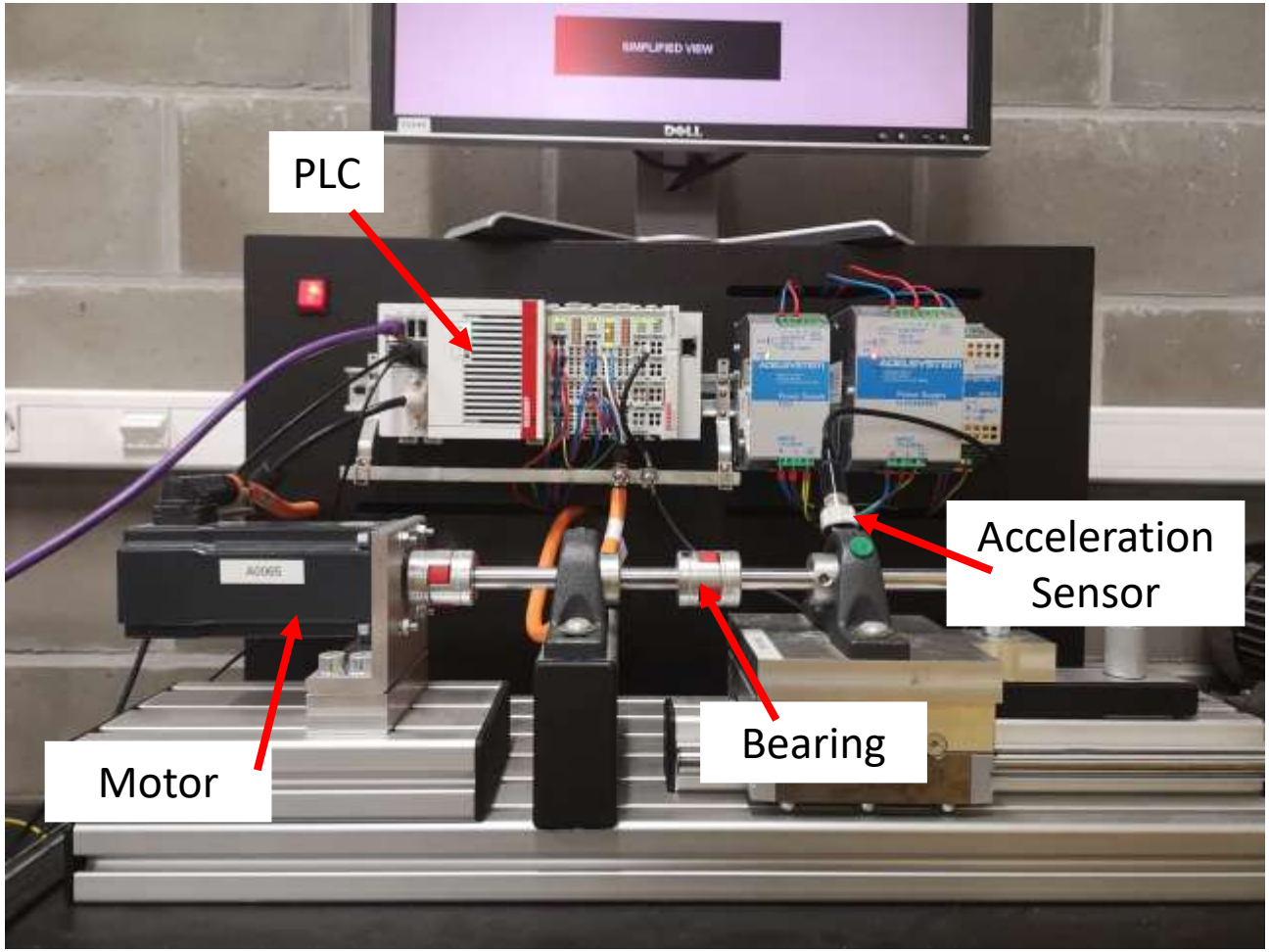
Cases and demonstrators

Demonstrator – Beckhoff motor

Lara Luys

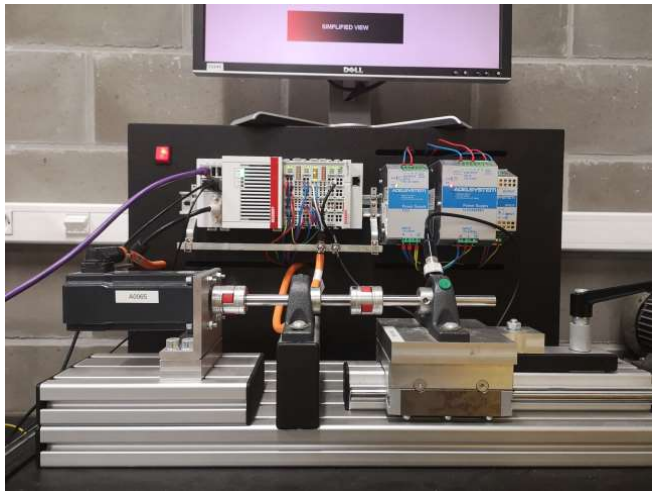
Demonstrator – Beckhoff motor

- Motor with bearings
- Acceleration sensor
- Controlled by PLC



Demonstrator – Beckhoff motor

- Goal:
- Detection of faulty bearings = classification model
- MLOps: detection of drift = anomaly detection
- Both models running on the PLC

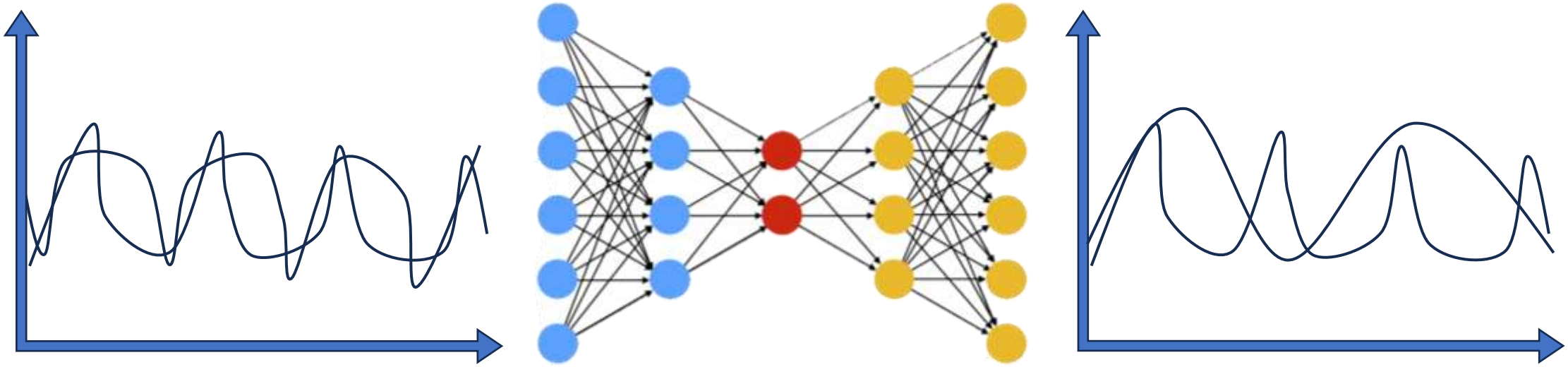


Demonstrator – Beckhoff motor

- Student project goal:
  - Creation of classification model
  - Model inference running on PLC
- What has been done:
  - Model creation: classification of different faults
- Next steps:
  - Model running on the PLC
  - Visualisation

Demonstrator – Beckhoff motor

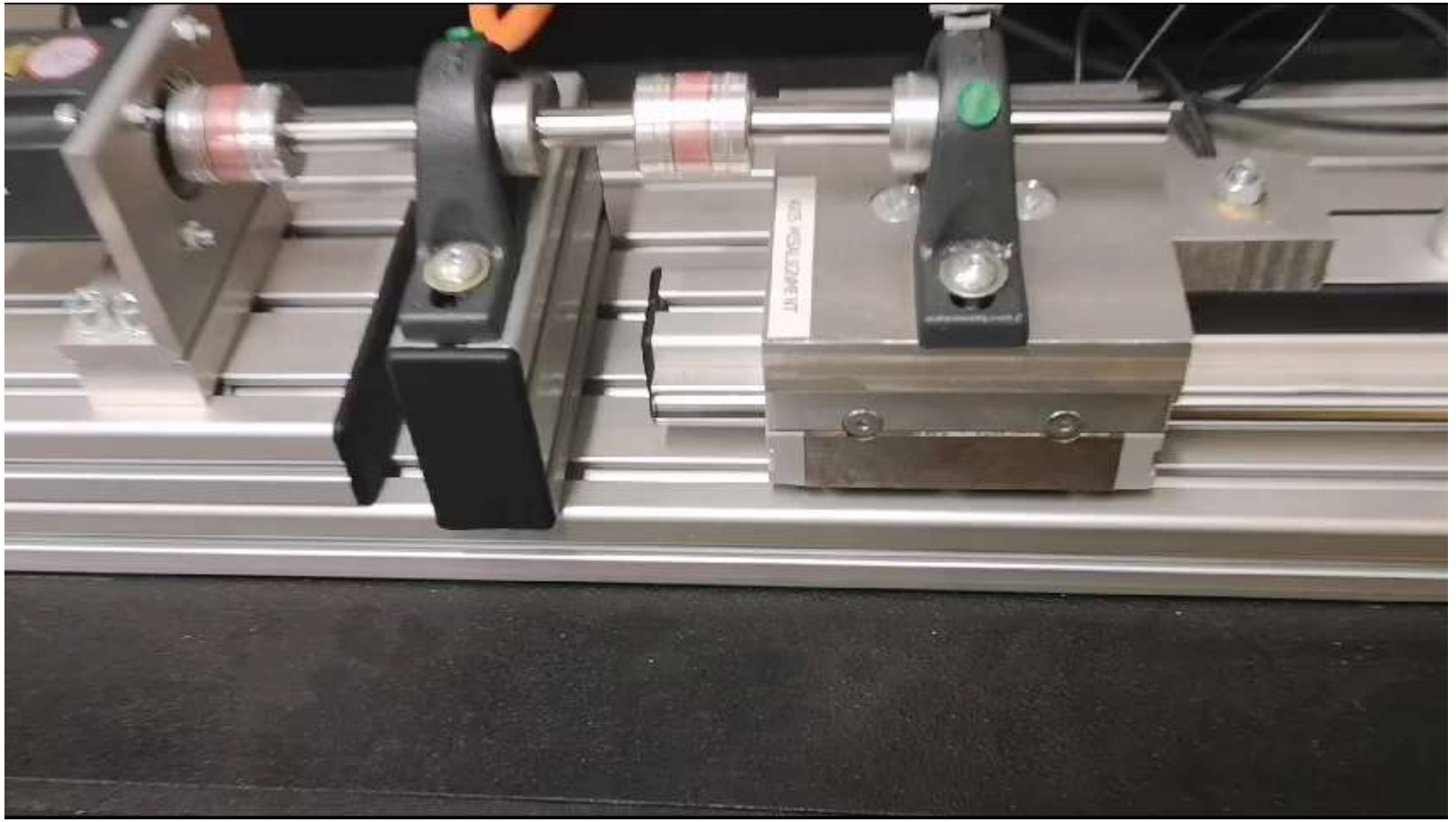
- Anomaly detection → autoencoder
- Autoencoder = recreates inputs
- If recreation is not correct → anomaly / drift detected



Demonstrator – Beckhoff motor

- Autoencoder goal:
  - Autoencoder creation on 1 type of data
  - Model on the PLC
- What has been done:
  - Creation of very simple autoencoder
  - Simple model running on PLC
- Next steps:
  - Create better autoencoder

Demonstrator Beckhoff motor



Demonstrator - Magic Wand

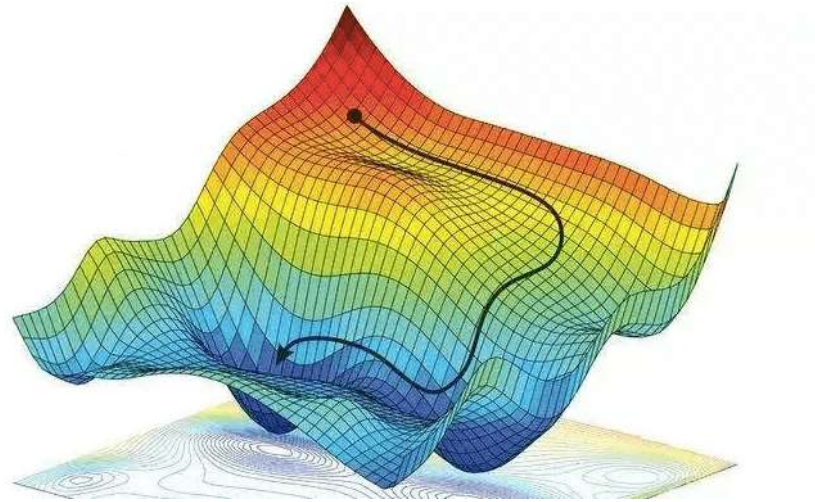
Alexander D’hoore

On-device learning - How?

How to learn on small devices?

- Naive approach: **gradient descent**

- Problems:
  - We can’t store entire dataset
  - Catastrophic forgetting (overfitting)
  - Needs more memory than inference
  - Probably too slow anyway

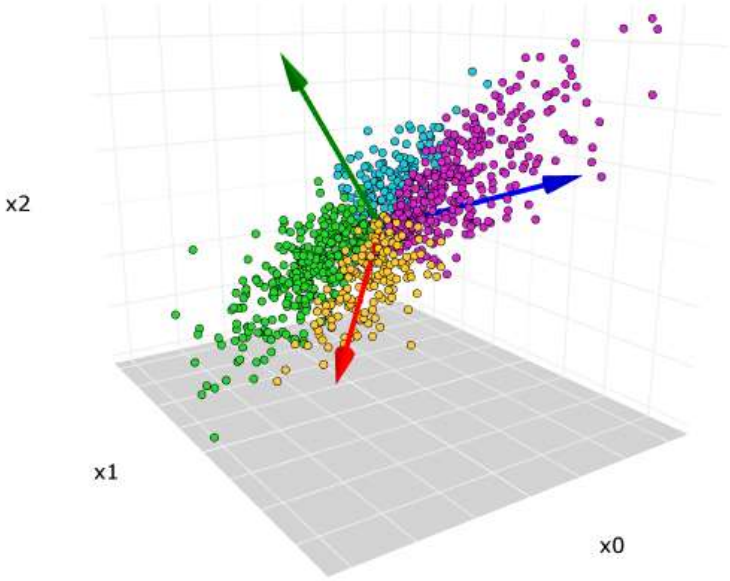


On-device learning - Embeddings

- Better approach: **embeddings!**
  - Dimensionality reduction technique
  - Encode long input into short output

- “Learn” things in embedding space
  - Classification using nearest-neighbor
  - Works for audio, video, sensor data…

- Much easier than gradient descent

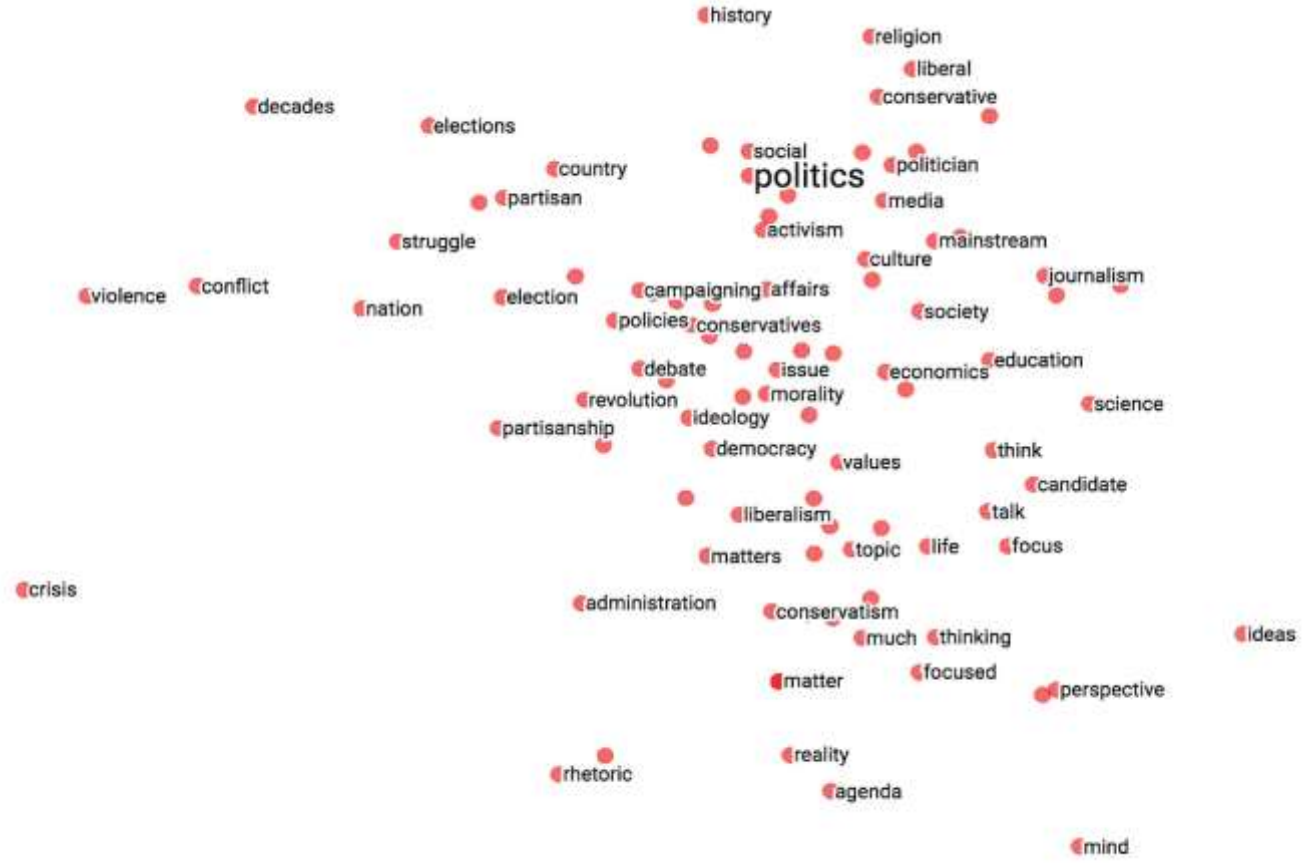


On-device learning - Embeddings

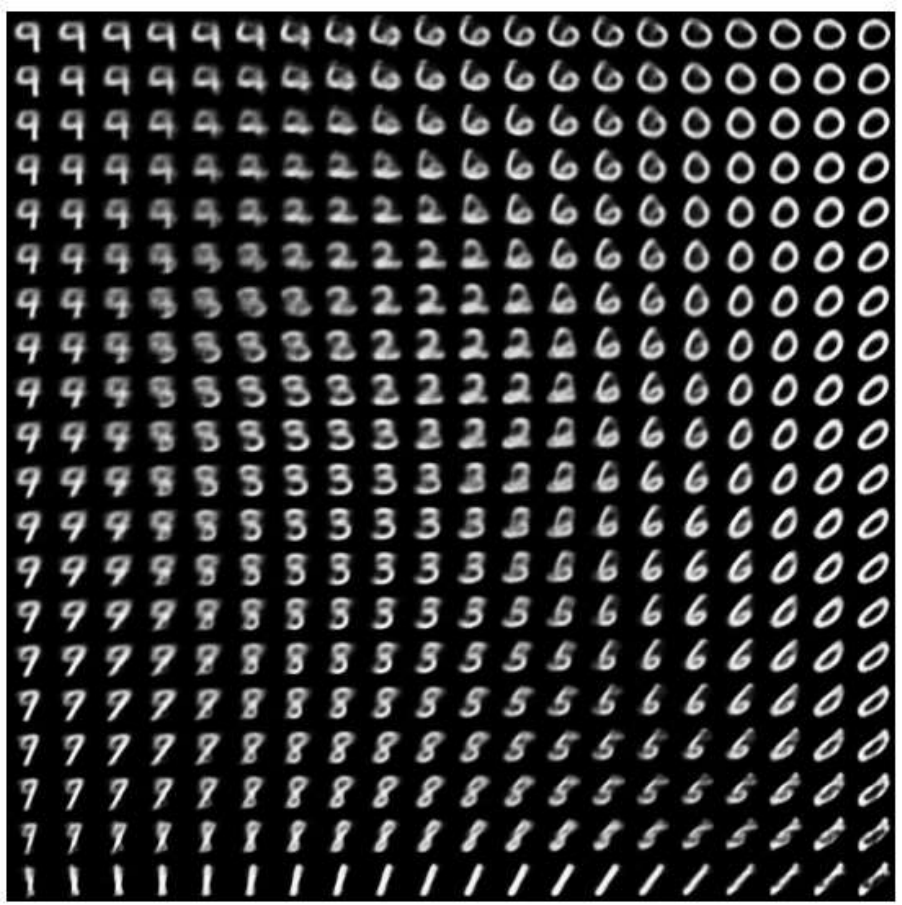
- How to train model:
  - Simply cut off classification head
  - Use activations from any middle layer
  - Unsupervised learning (autoencoders)

- Requirements:
  - Model must be trained on diverse dataset
  - Otherwise embeddings will be overfit,
  - and won’t be able to encode new inputs

On-device learning - Embeddings



On-device learning - Embeddings



Magic Wand - Intro

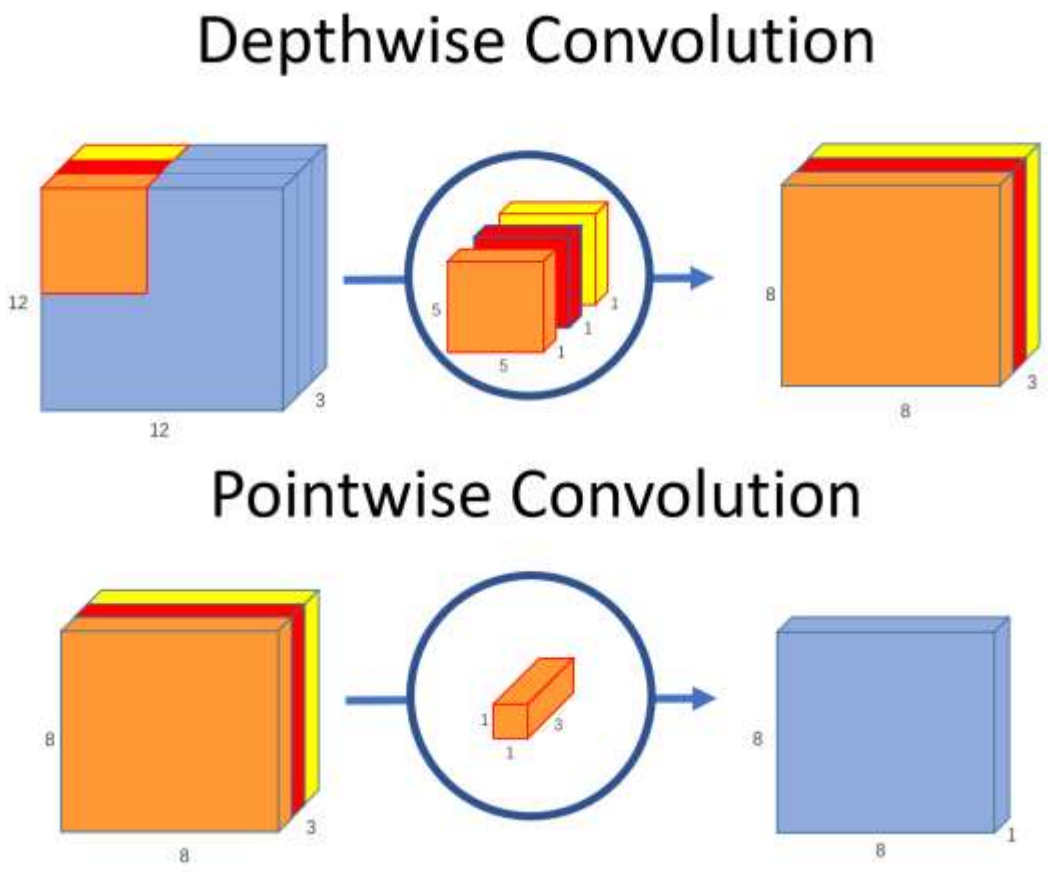
- Introducing: **Magic Wand!**

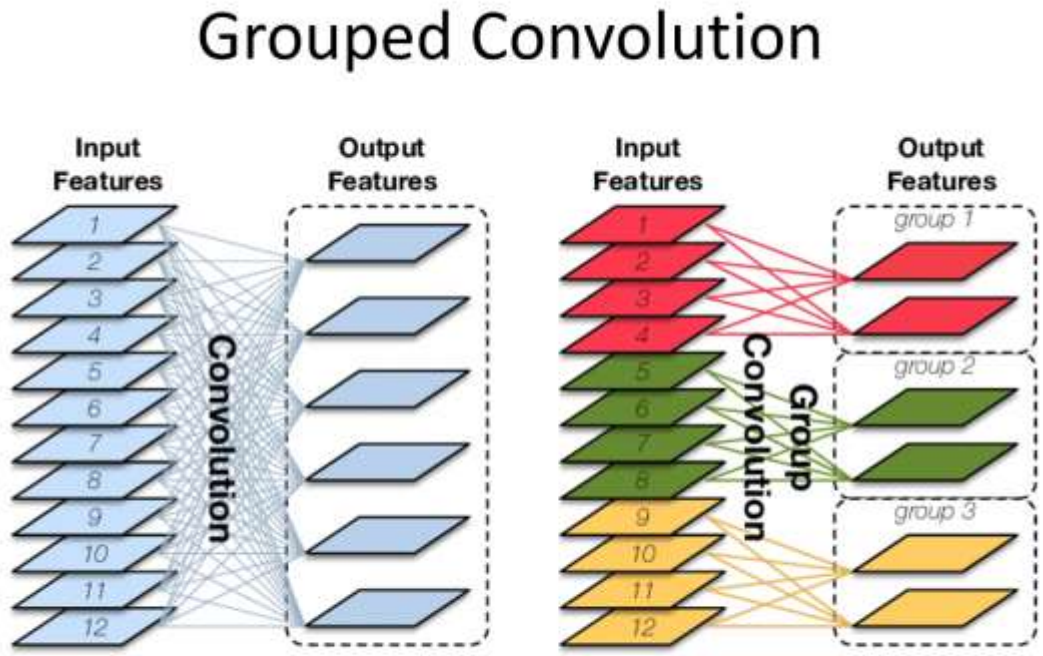
- Very small neural net on microcontroller
  - 1 MB Flash, 256 KB RAM, 64 MHz clock
  - Pre-trained on diverse set of gestures

- Used to classify gestures directly on MCU
  - System learns new gestures from examples

- **Can classify gestures it has never seen before**

Magic Wand - Architecture





Magic Wand Demo

Connect Wand

...

Magic Wand Demo

Connect Wand

...

Magic Wand - VIVES - G

OBS 30.1.2 - Profiel: Naa

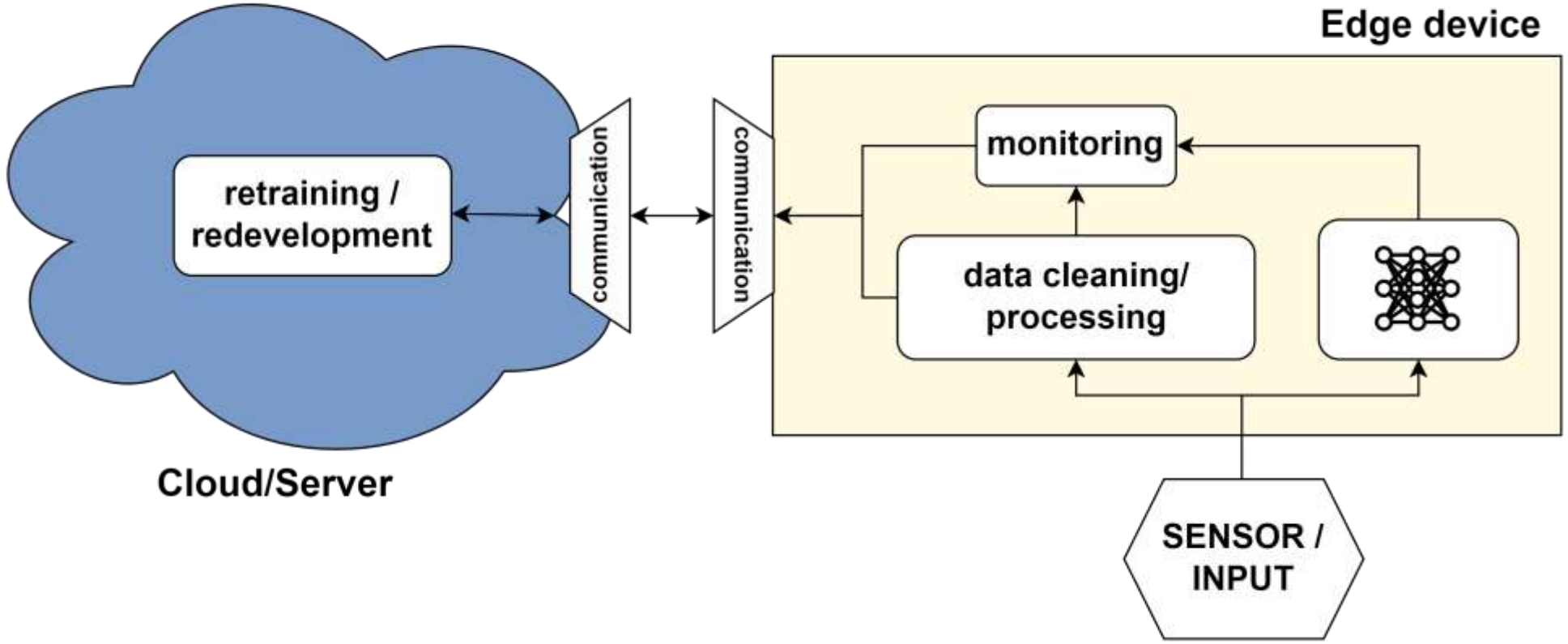
Use case – Vandewiele & Leap Technologies

Lara Luys

Use case – Vandewiele & Leap Technologies

- Machine learning model = fault detection
- Training data with different machine settings
- Different settings → prediction drift
- Goal
  - Run ML model on edge device
  - Monitoring on the edge
  - Retraining of the model on the cloud/server

Use case – Vandewiele & Leap Technologies



Use case – Vandewiele & Leap Technologies

- Workflow orchestration - Prefect
- Model deployment CI/CD - Gitlab
- Model and data monitoring – Evidently AI
- Model versioning/experiment tracking – MLflow
- Data versioning - DVC

Use case – Vandewiele & Leap Technologies

- What has been done:
  - Server set up with necessary tools
  - Data and model exploration
  - Monitoring setup in docker container simulating edge device
  - Evidently setup for data and prediction monitoring
  - Data split into reference data and drifted data
- Next steps:
  - Create model trained on data with constant machine settings
  - MLflow and DVC implementation
  - Selection of edge device
  - Evidently communication with server/cloud
  - Retraining of ML model on server/cloud

Volgende stappen

Voorstel workshops

- Full-stack Machine Learning (1 dag)
  - Leren werken met de tools (vooraf geïnstalleerd)
  - Data versioning, model tracking, monitoring
- MLOps Platform Engineering (1 dag)
  - De tools opzetten & configureren
  - DevOps for ML: CI/CD, docker, kubernetes
- MLOps on the Edge (1 dag)
  - Hardware platformen & tools (zie ONNX slides)
  - Deployment, upgrade & monitoring

Seminariedag MLOps4ECM

- Een halve dag sprekers uit de industrie
- Publieke event, toegankelijk voor externen
- Waarschijnlijk in najaar 2024

- Sprekers mogen zich aanmelden!

Case studies – Call to action!

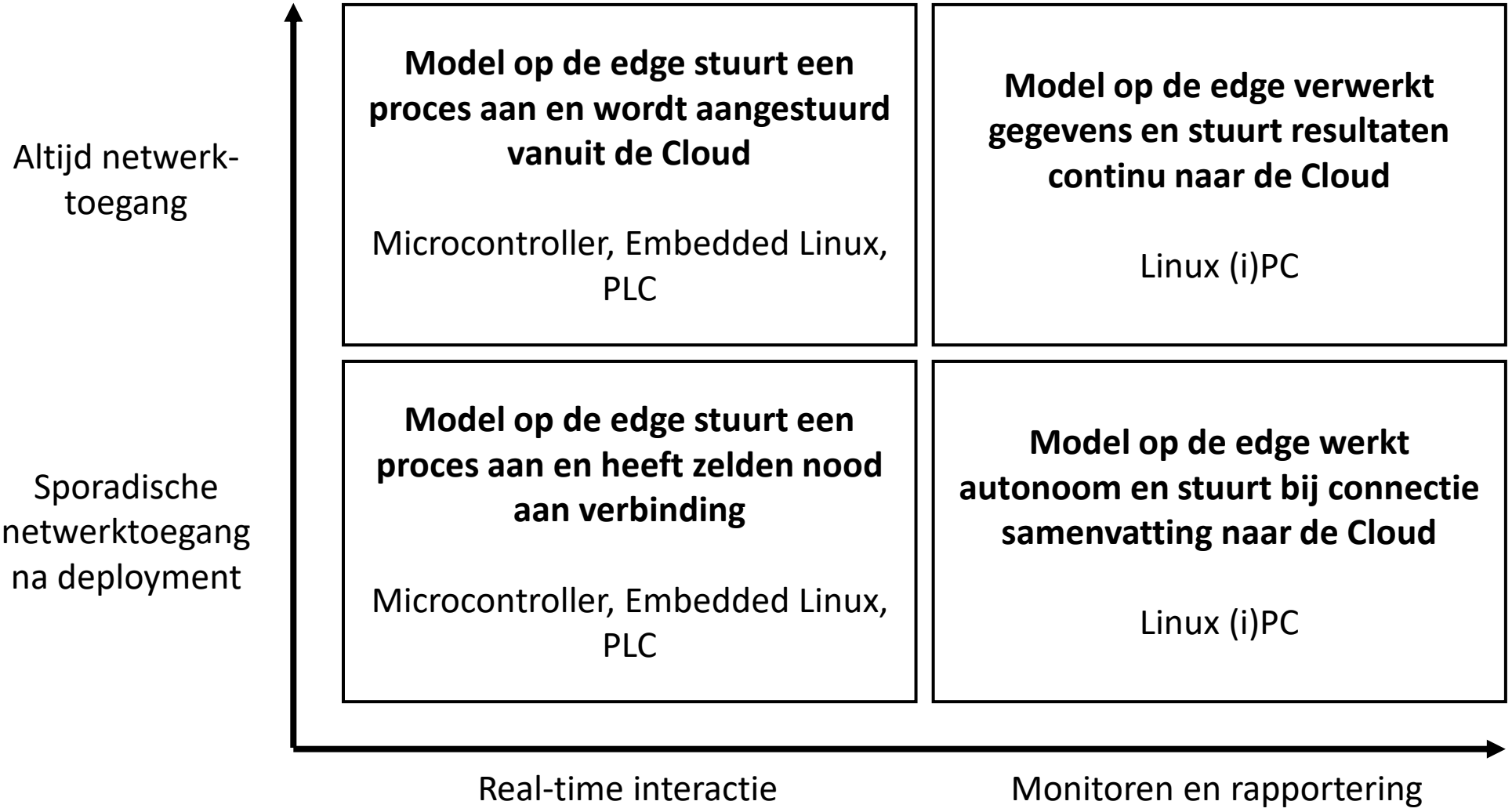
Belangrijke overwegingen:

- Beschikbaarheid van geannoteerde data en modellen
- Bereidheid tot actieve participatie vanuit het bedrijf
- Weinig tot geen beperkingen inzake confidentialiteit

Inspiratie: mogelijkheden - toepassingen

- Detecteren van fouten via trilling (motoren, lagers, wielen…)
- Spanning/stroom van motoren analyseren
- Productiecontrole aan de hand van visie (landbouw, bouw, voeding…)
- Monitoren van machine- en productkwaliteit via sensoren
- Voorspellen van onderhoud (machines, productielijnen…)

Inspiratie: mogelijke eisen/noden



Planning volgende periode

1. Rapportering aanwezige kennis, toepassingen
2. Verder uitwerken huidige cases & demonstratoren
3. Contact & opstart met bedrijf als partner voor use-case
4. Communicatie volgende vergadering van de Begeleidingsgroep
5. Website met login beschikbaar stellen met alle projectinformatie

Save the date – 14/11/2024

~~• Niet 13 november!~~ **Wel 14 november**

- Next usergroup meeting

- Location?

Feedback - vragenlijst

Beschikbaar via

https://forms.office.com/e/MzARxJWLGB

Binnenkort ook via https://mlops4ecm.be

VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Tussentijdse vergadering 16/05/2024

Locatie: Vandewiele nv

Met steun van VLAIO

HBC.2023.0062