Uitvoeringsorganisatie
Bedrijfsvoering Rijk
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

MLOpsing on open data of Groningen

Using MLflow and DVC to build a robust ML system

Laurens Weijs | Data Engineer Rijks ICT Gilde

About me

Econometrics (Quantitative Finance) @ Rotterdam

Computer Science (Software Technology) @ Delft

Data engineer @ ML6 - Google Cloud Partner

Data engineer @ Rijks ICT Gilde

Laurens Weijs
Data Engineer Rijks ICT Gilde

Agenda

Goal of this presentation

Introducing MLOps

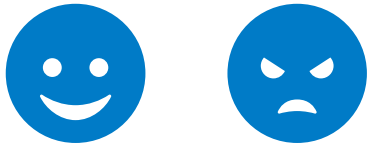
Dataset.dvc

Requirements.txt

Applying MLOps on Open data of Groningen

Goal

- Get an insight in the sentiment of the people of Groningen



- By using Open data
- By tracking our Model & Data & Code

Agenda

Goal of this presentation

Introducing MLOps

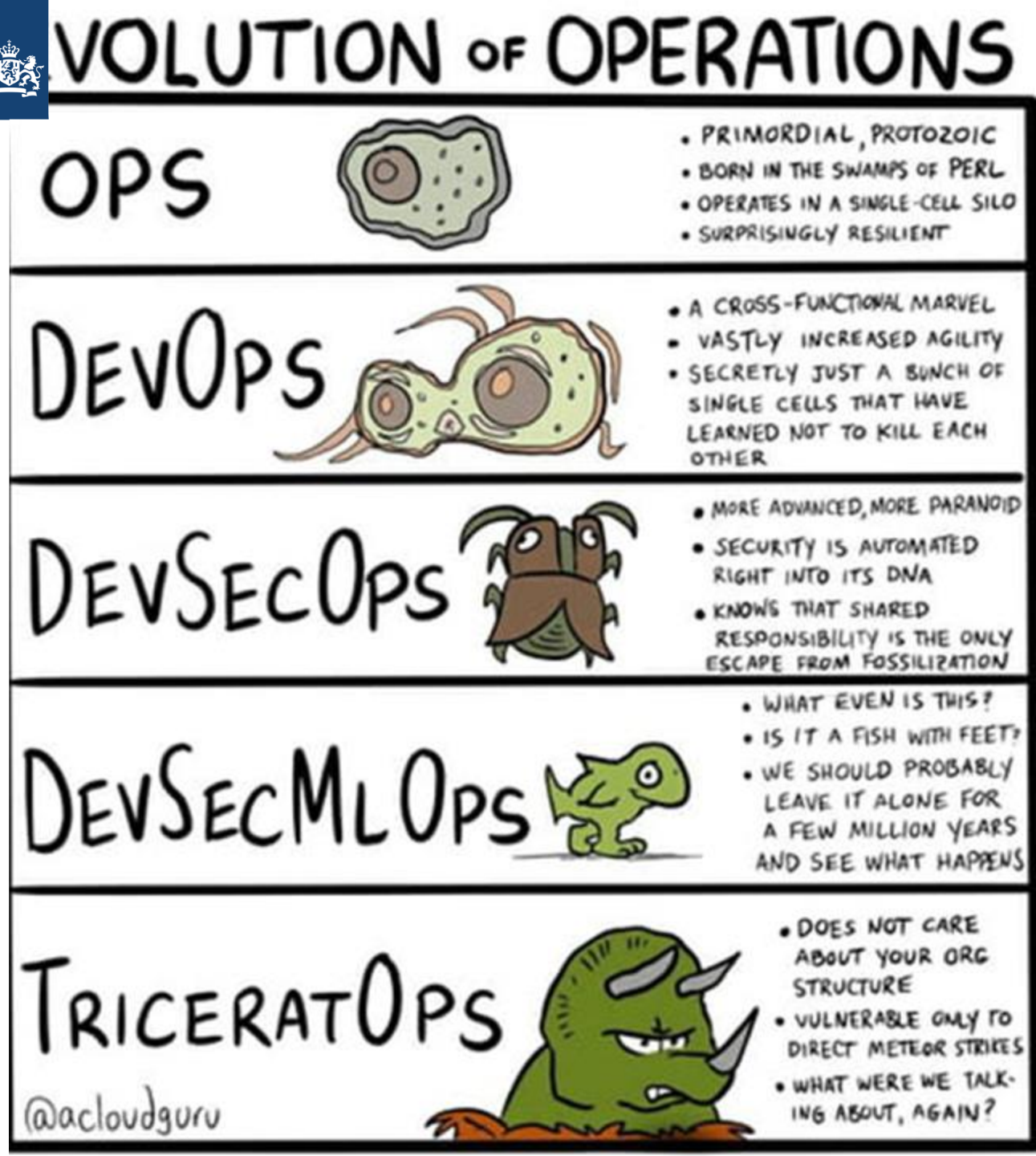
Dataset.dvc

Requirements.txt

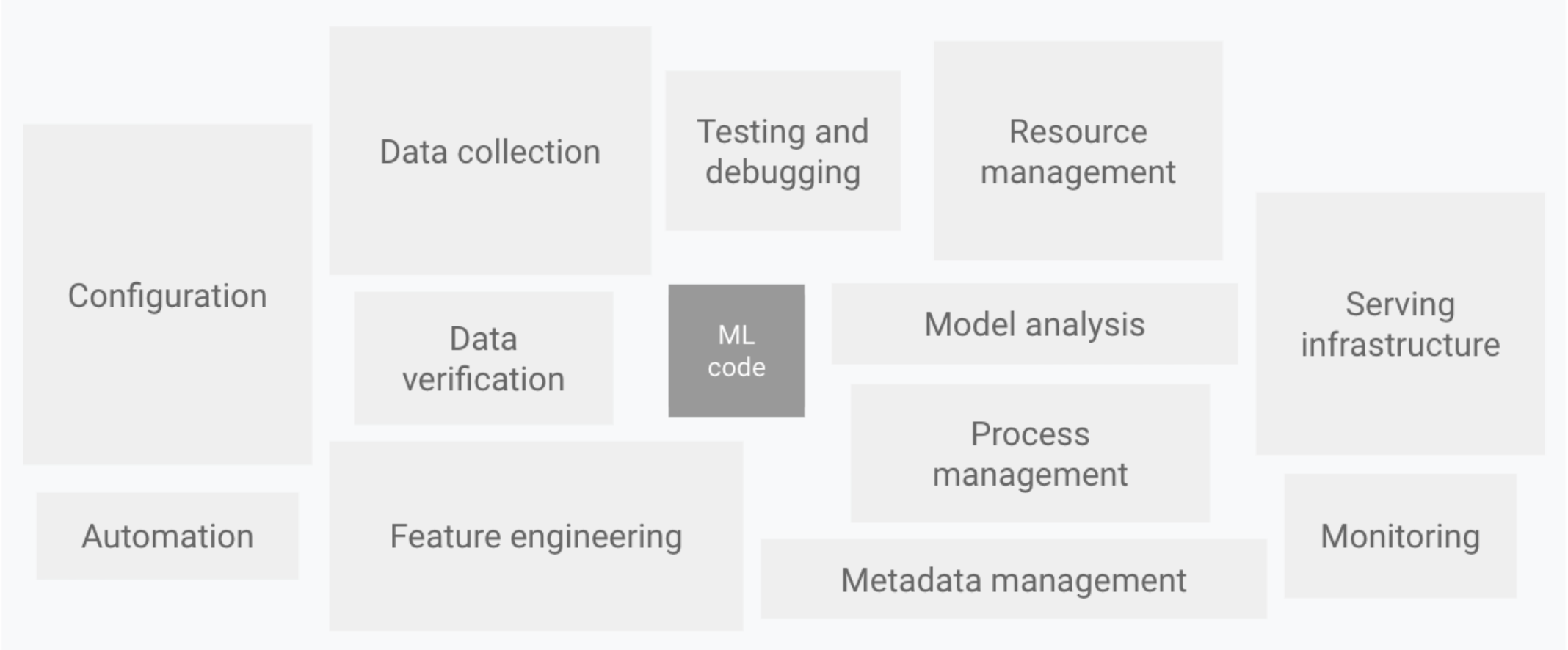
Applying MLOps on Open data of Groningen

What is MLOps?

- Machine Learning + Operations = MLOps
- DevOps for Machine Learning systems
- MLOps – Wikipedia - Maintaining of production ML lifecycle (software deployment, CI/CD, Orchestratie, Data Health).



ML code is just a small part of a ML System



"The hidden technical debt of Machine Learning Systems"

ML systems operate in 3 dimensions

Data	Model	Code
Schema	Algorithms	Business Needs
Sampling over Time	More Training	Bug Fixes
Volume	Experiments	Configuration

Agenda

Goal of this presentation

Introducing MLOps

Dataset.dvc

Requirements.txt

Applying MLOps on Open data of Groningen

Dataset.dvc

- **Location?**

  Groningen data platform

  https://groningen.dataplatform.nl/#/data/885dbda5-efc1-4d64-b1d6-8444cd5a4cb8

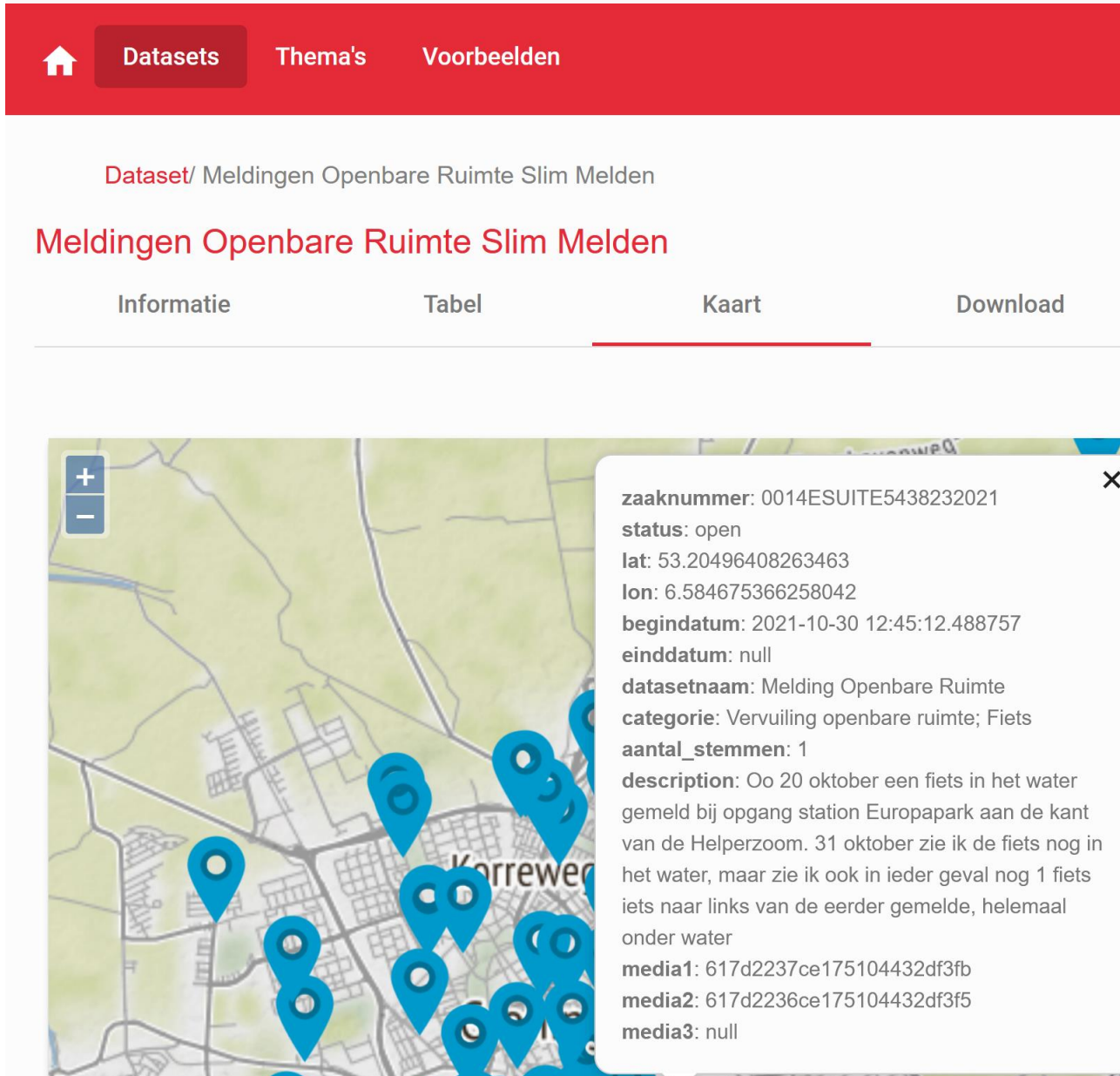
- **What is it about?**

  Remarks from citizens about public spaces from any citizen with the App

Dataset.dvc

There is still a bike in the canal, already from the 20th of october to the 31th of October.

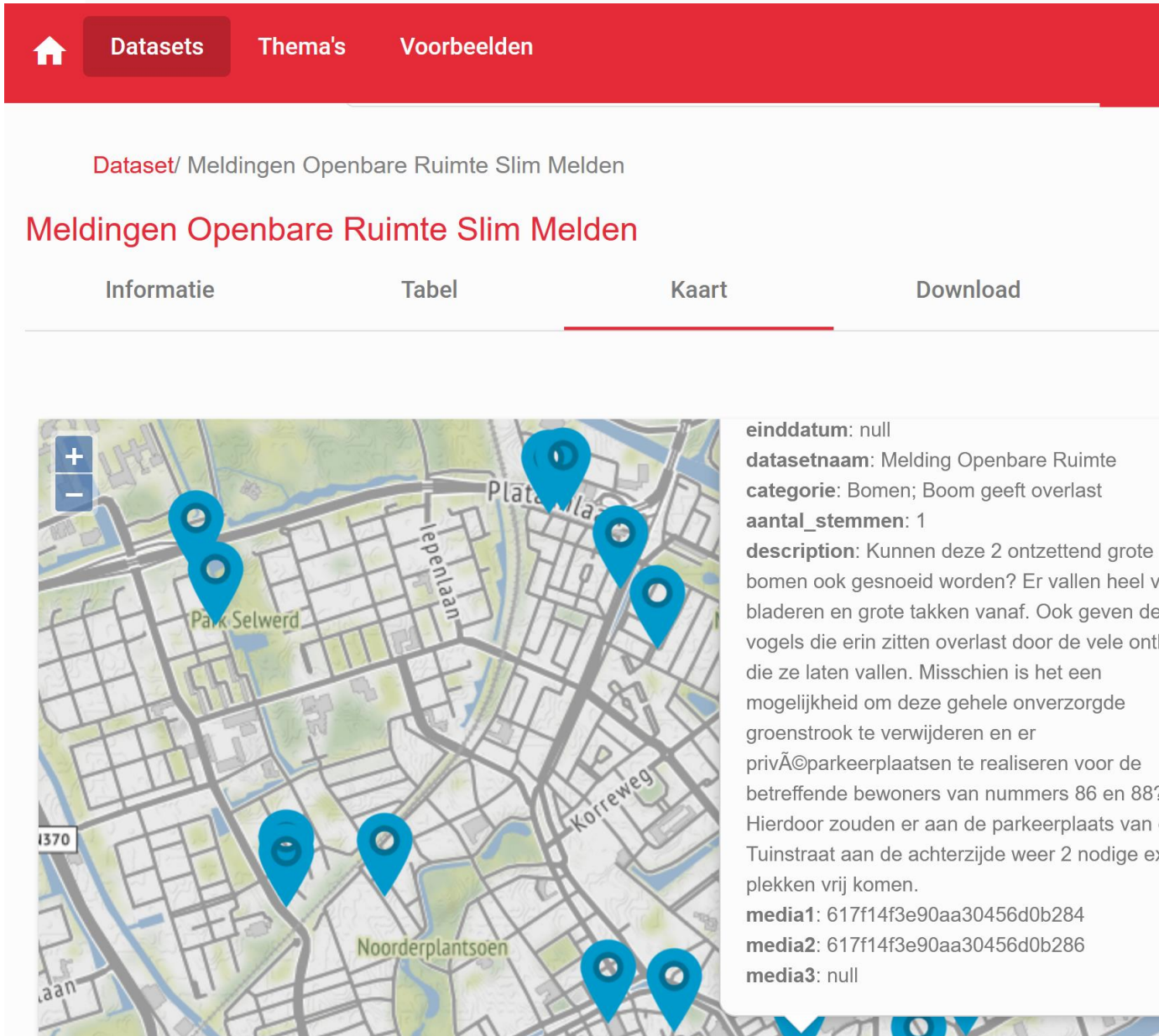
There seems to be an extra bike in the canal as well.



Dataset.dvc

Can you remove the two large trees in front of my house as they are annoying with leaves and branches.

Also, could you make two private parking places there then?



Agenda

Goal of this presentation

Introducing MLOps

Dataset.dvc

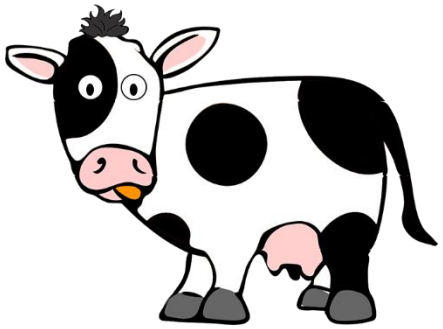
Requirements.txt

Applying MLOps on Open data of Groningen

Requirements.txt

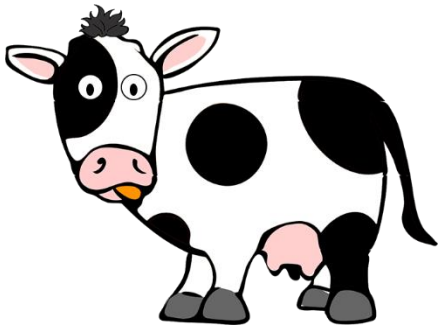
- Runner: **Python** ☺

Requirements.txt



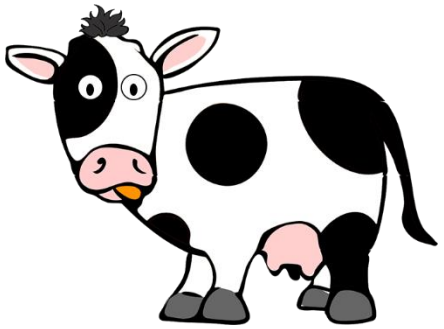
- Runner: **Python** ☺
- Model: **Bertje**
  - ***https://huggingface.co/GroNLP/bert-base-dutch-cased***

Requirements.txt



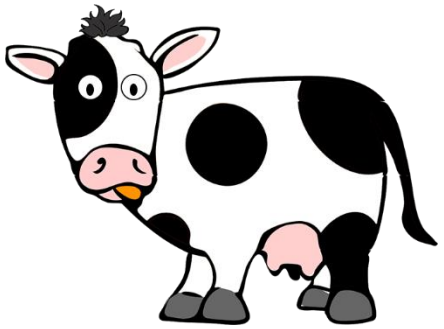
- Runner: **Python** ☺
- Model: **Bertje**
  - ***https://huggingface.co/GroNLP/bert-base-dutch-cased***
- ML Framework: **Pytorch**

Requirements.txt



- Runner: **Python** ☺
- Model: **Bertje**
  - ***https://huggingface.co/GroNLP/bert-base-dutch-cased***
- ML Framework: **Pytorch**
- Git for data: **DVC**

Requirements.txt



- Runner: **Python** ☺
- Model: **Bertje**
  - ***https://huggingface.co/GroNLP/bert-base-dutch-cased***
- ML Framework: **Pytorch**
- Git for data: **DVC**
- Experiment tracking: **mlflow**

Agenda

Goal of this presentation

Introducing MLOps

Dataset.dvc

Requirements.txt

Applying MLOps on Open data of Groningen

Applying MLOps on Open data of Groningen

If you want to play along with yourself go to: https://github.com/RIG-MYCELIA/pygrunn_2021_mlops

Problem:

Apply sentiment analysis on the remarks on the public places, while keeping track of the data and the experiments with DVC and MLflow.

Steps

- **1: Download dataset & init DVC**
- **2: Run experiment with model**
- 3: Manually label a few features
- 4: Fine-tune the model & Run experiment with new model
- 5: Change back the dataset
- 6: Run experiment with old data and new model

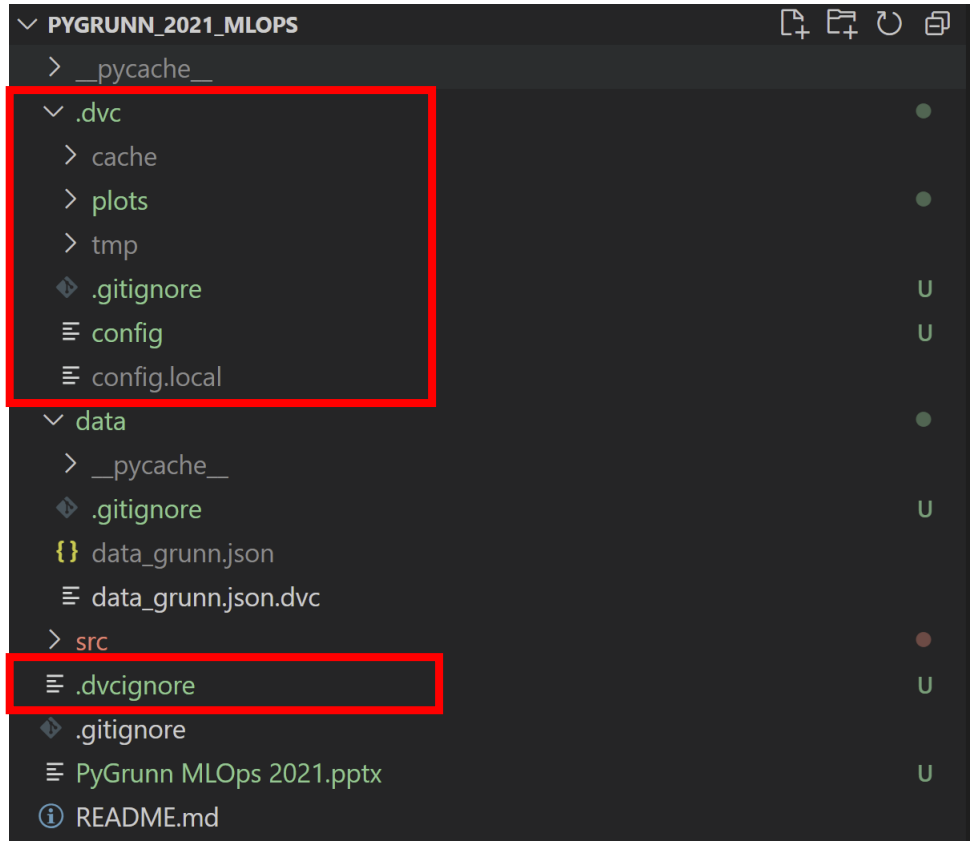
Downloading the dataset – The code

```python
import urllib3
import json

# REST call to the Data platform of Groningen
http = urllib3.PoolManager()
resource_id_grunn = 'edc8db66-8dc3-4819-9935-e6c8f55388ab'
limit = 2000
url = 'https://ckan.dataplatform.nl/api/action
/datastore_search?limit={}&resource_id={}'.format(limit,
resource_id_grunn)
response = http.request('GET', url)
data = json.loads(response.data)

# Write away the open data to a json file
with open('data/data_grunn.json', 'w') as fp:
    json.dump(data, fp)
```

Initializing DVC – The code

```
# Initialize DVC in the current folder
dvc init

# Set the remote repository for DVC
dvc remote add -d azure_storage azure://dvc
dvc remote modify --local azure_storage connection_string SECRET_KEY
```

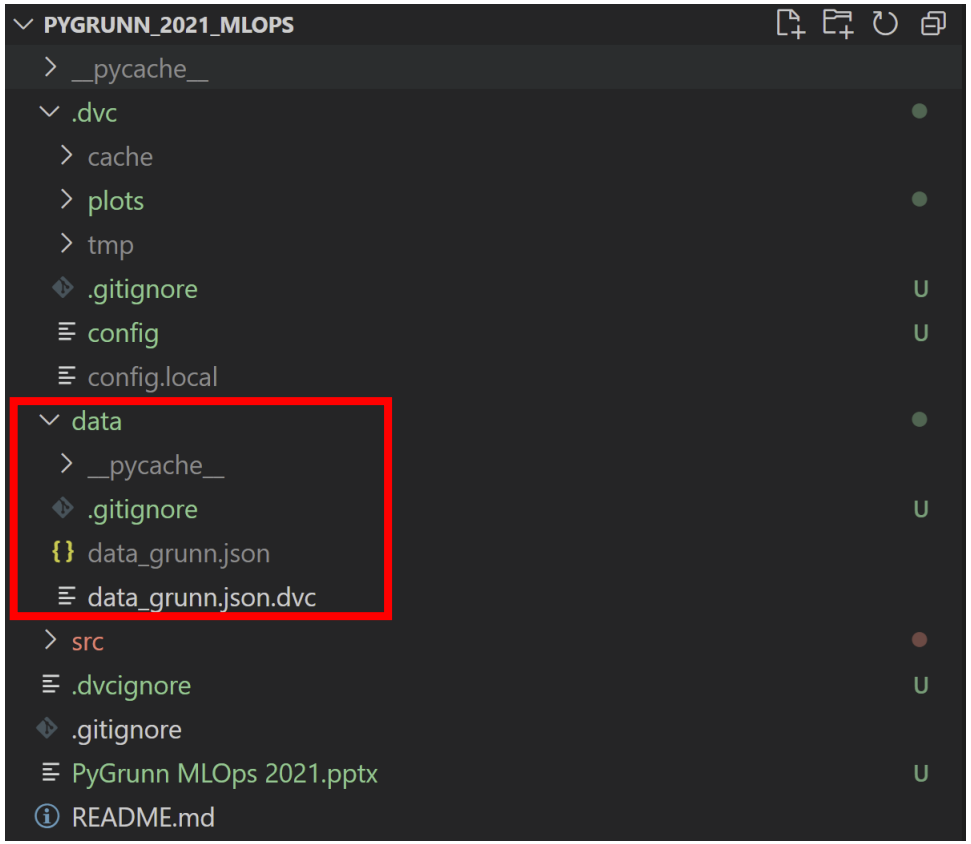
Initializing DVC – The folder structure



Committing the data – The code

```
# Add the data for tracking
dvc add data/data_grunn.json

# Add the dvc files to git
git add data/data_grunn.json.dvc
git add data/.gitignore
git commit -m "dvc data tracking"
git push
```

Committing the data – The folder structure



Committing the data

pygrunn_2021_mlops / data / data_grunn.json.dvc

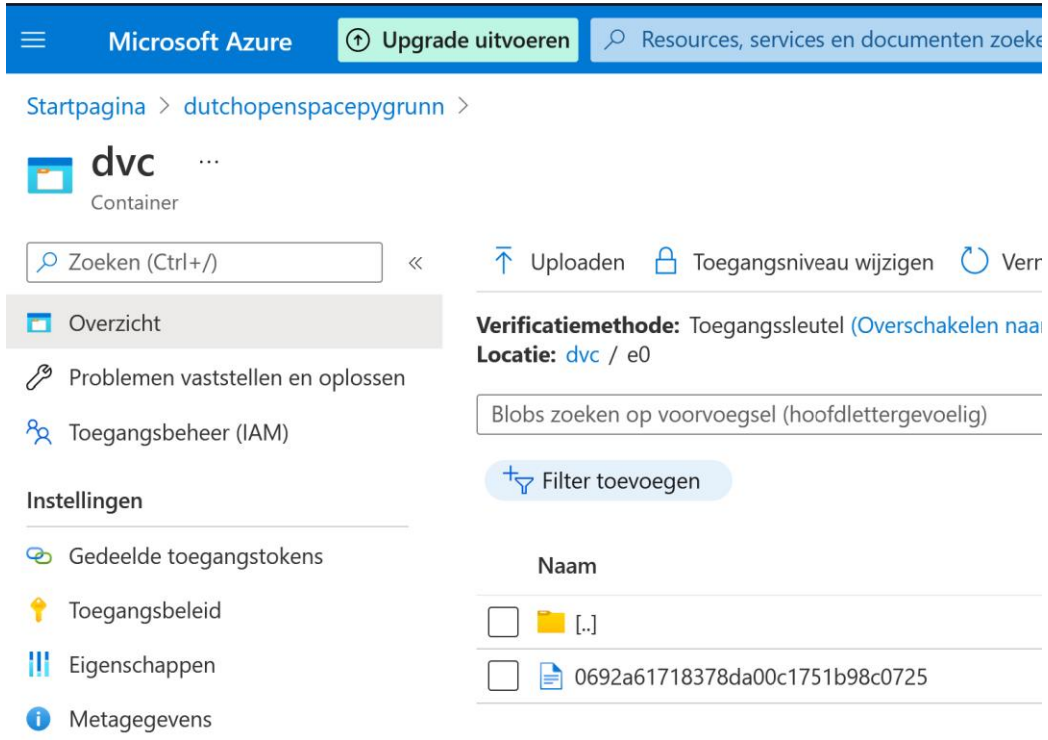
Laurens Weijs base python files, initial commit

0 contributors

4 lines (4 sloc) | 84 Bytes

```
outs:
- md5: e00692a61718378da00c1751b98c0725
  size: 52879
  path: data_grunn.json
```

Reference of the data on git



Actual data in blob storage

Steps

- 1: Download dataset & init DVC
- **2: Run experiment with model**
- 3: Manually label a few features
- 4: Fine-tune the model & Run experiment with new model
- 5: Change back the dataset
- 6: Run experiment with old data and new model

Wrapper class for MLflow – The code

```python
# Wrapper class for the hugging face sentiment analysis pipeline in MLflow pyfunc
class SentimentAnalysis(mlflow.pyfunc.PythonModel):
    '''
    Any MLflow Python model is expected to be loadable as a python_function model.
    '''

    def __init__(self):
        from transformers import (AutoModelForSequenceClassification,
                                  AutoTokenizer)

        huggingface_hub_model_name = "GroNLP/bert-base-dutch-cased"
        self.tokenizer = AutoTokenizer.from_pretrained(huggingface_hub_model_name)
        self.sentiment_analysis = AutoModelForSequenceClassification.from_pretrained(huggingface_hub_model_name)

    def predict(self, model_input):
        classifier = pipeline('sentiment-analysis', model=self.sentiment_analysis, tokenizer=self.tokenizer)
        model_input['name'] = model_input['text'].apply(classifier)

        return model_input
```

Logging our model with MLflow – The Code

```python
import mlflow
from mlflow.models import ModelSignature

from sentiment_analysis import SentimentAnalysis

# Input and Output formats
input = json.dumps([{'name': 'text', 'type': 'string'}])
output = json.dumps([{'name': 'text', 'type': 'string'}])
# Define Signature for the mlflow model to define the input and output
signature = ModelSignature.from_dict({'inputs': input, 'outputs': output})

# Start tracking
with mlflow.start_run(run_name="groningen_sentiment_analysis") as run:
    print(run.info.run_id)
    runner = run.info.run_id
    mlflow.pyfunc.log_model('model', python_model=SentimentAnalysis(),
                            signature=signature)
```

Starting up the UI of MLflow – The code

```
# Starting up the UI of MLflow
mlflow ui
INFO:waitress:Serving on http://127.0.0.1:5000
```

Starting up the UI of MLflow – The UI

mlflow Experiments Models GitHub Docs

Experiments

Search Experiments

Default

Default

Track machine learning training runs in an experiment. Learn more

Experiment ID: 0

Notes

Showing 50 matching runs

Refresh | Compare | Delete | Download CSV | Start Time | All

Columns | Only show differences | metrics.rmse < 1 and params.model = "tree" | Search | Filter | Clear

							Metrics >		
Start Time	Duration	Run Name	User	Source	Version	Models	epoch	total_flos	trai
1 day ago	10.1s	groningen_s...	Laurens RIG	log_model_	574022	pyfunc	-	-	
1 day ago	4.0s	-	Laurens RIG	finetune_se	62c9ca	-	3	5796665438...	
1 day ago	352ms	-	Laurens RIG	finetune_se	62c9ca	-	-	-	
1 day ago	4.2s	-	Laurens RIG	finetune_se	62c9ca	-	3	5796665438...	

Inspecting the Logged model – The UI

Artifacts

- model
  - MLmodel
  - conda.yaml
  - python_model.pkl
  - requirements.txt

Full Path:file:///C:/Users/Laurens%20RIG/Documents/pygrunn_2021_mlops/mlruns/0/9c9662d0adbb4e7faf034a00...

Register Model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control

Model schema

Input and output schema for your model. Learn more

Name	Type
Inputs (1)	
text	string
Outputs (1)	
text	string

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/9c9662d0adbb4e7faf034a0047df27bb/model'

# Load model as a Spark UDF.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)

# Predict on a Spark DataFrame.
columns = list(df.columns)
df.withColumn('predictions', loaded_model(*columns)).collect()
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/9c9662d0adbb4e7faf034a0047df27bb/model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

Making a prediction – The Code

```python
# Load in the data
filename = 'data/data_grunn.json'
with open(filename, 'r') as f:
  descriptions = json.loads(f.read())

# Load the model as a PyFuncModel.
mlflow_run_id='runs:/9c9662d0adbb4e7faf034a0047df27bb/model'
loaded_model = mlflow.pyfunc.load_model(mlflow_run_id)

# Predict on a Pandas DataFrame.
res = loaded_model.predict(pd.DataFrame(descriptions))
with open(output_filename, 'w') as f:
    f.write(json.dumps(res.to_json()))
```

Making a prediction – The Results

	text	name
0	Losliggende tegels. Erg gevaarlijk in het donker. Aantal mensen al gestruikeld.	[{'label': 'neg', 'score': 0.9996267557144165}]
1	Zeer donker pad naar kdv, school Ã©n bso, graag extra straatverlichting aan dit pad.	[{'label': 'pos', 'score': 0.9998181462287903}]
2	Deze straatverlichting is kapot	[{'label': 'neg', 'score': 0.9998509883880615}]



Now go make the people of Groningen more happy by tracking your data and models!

Questions?

laurens.weijs@rijksoverheid.nl

github.com/RIG-MYCELIA/pygrunn_2021_mlops