# Infrastructure in ML projects

23.01.2020

William Naylor

inmeta

# Pre-ramble

- The world hasn't worked out how to do ML
  - ML has been packaged in with IT
  - Many new products, beta access, and idiotic ideas (for you)
- We work on something like
  - custom solutions
  - python code
- You aren't really 100% sure of things in an ML model

inmeta

# Contents

- This talk will not include any fancy ML models
- What you need to get right in ML
- Two examples
  - PrettyPoly (AkerBP)
  - Sparebank 1 Kredittkort


inmeta


Things to get right
inmeta

# Data

- Match of data to problem
  - Fight for labels
- Understanding of data
- Ability to recall data from any time to any time
  - Should be able to reproduce old models
- Data (retraining) strategy going forward

```
dvdrental=# select title, release_year, length, replacement_cost from film
dvdrental-#   where length > 120 and replacement_cost > 29.50
dvdrental-#   order by title desc;
          title           | release_year | length | replacement_cost
--------------------------+--------------+--------+------------------
 West Lion                |         2006 |    159 |            29.99
 Virgin Daisy             |         2006 |    179 |            29.99
 Uncut Suicides           |         2006 |    172 |            29.99
 Tracy Cider              |         2006 |    142 |            29.99
 Song Hedwig              |         2006 |    165 |            29.99
 Slacker Liaisons         |         2006 |    179 |            29.99
 Sassy Packer             |         2006 |    154 |            29.99
 River Outlaw             |         2006 |    149 |            29.99
 Right Cranes             |         2006 |    153 |            29.99
 Quest Mussolini          |         2006 |    177 |            29.99
 Poseidon Forever         |         2006 |    159 |            29.99
 Loathing Legally         |         2006 |    140 |            29.99
 Lawless Vision           |         2006 |    181 |            29.99
 Jingle Sagebrush         |         2006 |    124 |            29.99
 Jericho Mulan            |         2006 |    171 |            29.99
 Japanese Run             |         2006 |    135 |            29.99
 Gilmore Boiled           |         2006 |    163 |            29.99
 Floats Garden            |         2006 |    145 |            29.99
 Fantasia Park            |         2006 |    131 |            29.99
 Extraordinary Conquerer  |         2006 |    122 |            29.99
 Everyone Craft           |         2006 |    163 |            29.99
 Dirty Ace                |         2006 |    147 |            29.99
 Clyde Theory             |         2006 |    139 |            29.99
 Clockwork Paradise       |         2006 |    143 |            29.99
 Ballroom Mockingbird     |         2006 |    173 |            29.99
(25 rows)
```

# Structured in what you are learning (scientific)

- When building ML models try and treat each ‘step’ like a scientific experiment
  - Make a hypothesis, write it down
  - Build and test the idea
  - If results aren’t as you predict find out why
  - Keep a record of experiments

inmeta

# Track your model in production

- You will want to see how your model performs over time
- You might have to roll back to a previous model

- Metrics


inmeta

# Some “software stuff”

- Git
  - Clean notebooks before commiting
- Linter
- Tests

```
def password_filter(text):
    '''
    There is no 'password' in the smartfilter terms so this
    should cover all cases where a password is given (in an
    english text).
    '''
    ex = r'''(pass( ?words?|wd)(([\sa-z0-9]{,3}:|[\sa-z0-9]{,3}=|(\s{1,3}(is|are)[^a-z]))|((\s{1,3}| ?(-|–|:|=) ?)((\S*[a-z]+\S*[\d]+\S*)|(\S*\d+\S*[a-z]+\S*)))|(\s{,3

    if re.search(ex, text, re.IGNORECASE | re.DOTALL) is None:
        return 0
    else:
        return 1
```

# Limit complexity

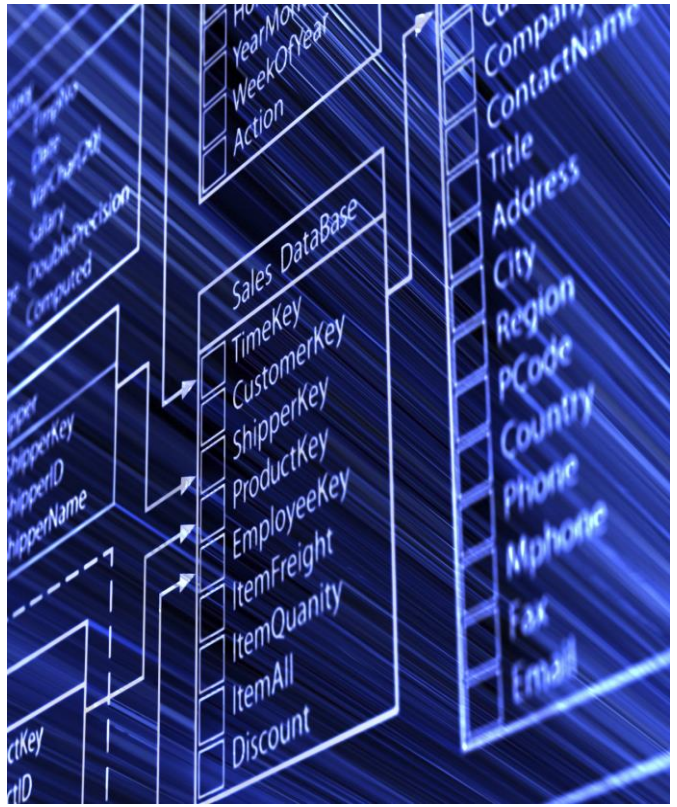
# Things to get right

1. Data
2. Structure in your work habits
3. Tracking model performance, and ability to go back to old models
4. Software stuff
5. Limit complexity


Sparebank 1
inmeta

# ML at SB1 Kredittkort

- Overview:
  - Team of people, including a DS, but good BI support and database
  - Have built a few different ML models, one of which is in use
- Data:
  - A lot of data comes in daily
  - All structured data.
  - On prem




inmeta

# The implementation

## test

SQL Server

- Anaconda
- Git repo
- Set of Cron jobs
- ML Config file

## prod

SQL Server

- Anaconda
- Git repo
- Set of Cron jobs
- ML Config file

git pull

git pull

## workstation (s)

- Anaconda
- Git repo
- ML Config file

git push

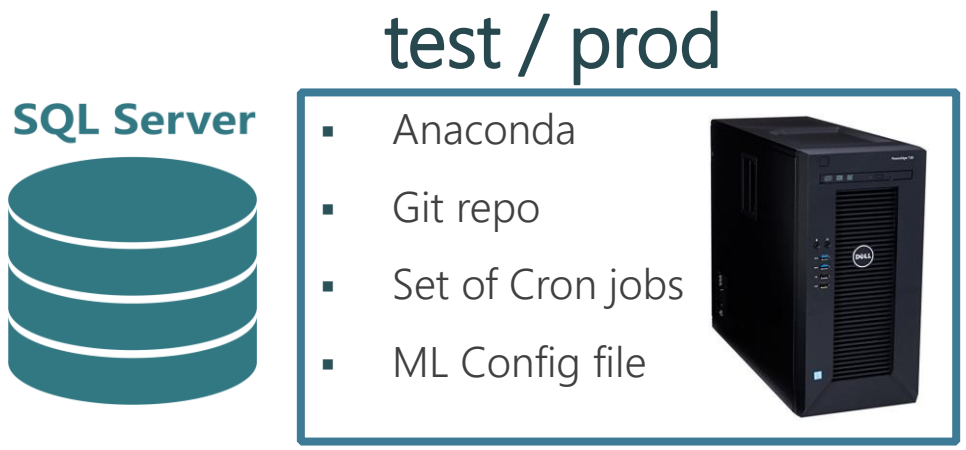
git pull

## git repo

inmeta

# What happens in test / prod

- Cron jobs run python scripts that are identical in test / prod
- Python scripts can grab variable that differ between test / prod from the ML config file.
- Training data is pulled in from the appropriate SQL server
- Seperate scripts for train / predict / score
- Various data / models / output is output to the PC file system, some data sent back to appropriate SQL server




inmeta

# AB testing

- Want to be able to say if model is working at all
- Test models against one another

- **train.py** creates model files:
  - modelA_v1.0_DATE.pkl and modelB_v1.0_DATE.pkl (and latest)
- **predict.py** loads modelA and modelB candidates
  - splits candidates
  - predicts using the models
  - write prediction to SQL table
  - write model name as string to SQL table

inmeta

# Notes and remarks

- Data
  - Was already in place (from BI team)
- Model accuracy and usefulness a part of system
- Very simple setup
- Scheduled response (not real time)
- No internet on prod computer made updating python libraries cumbersome

inmeta

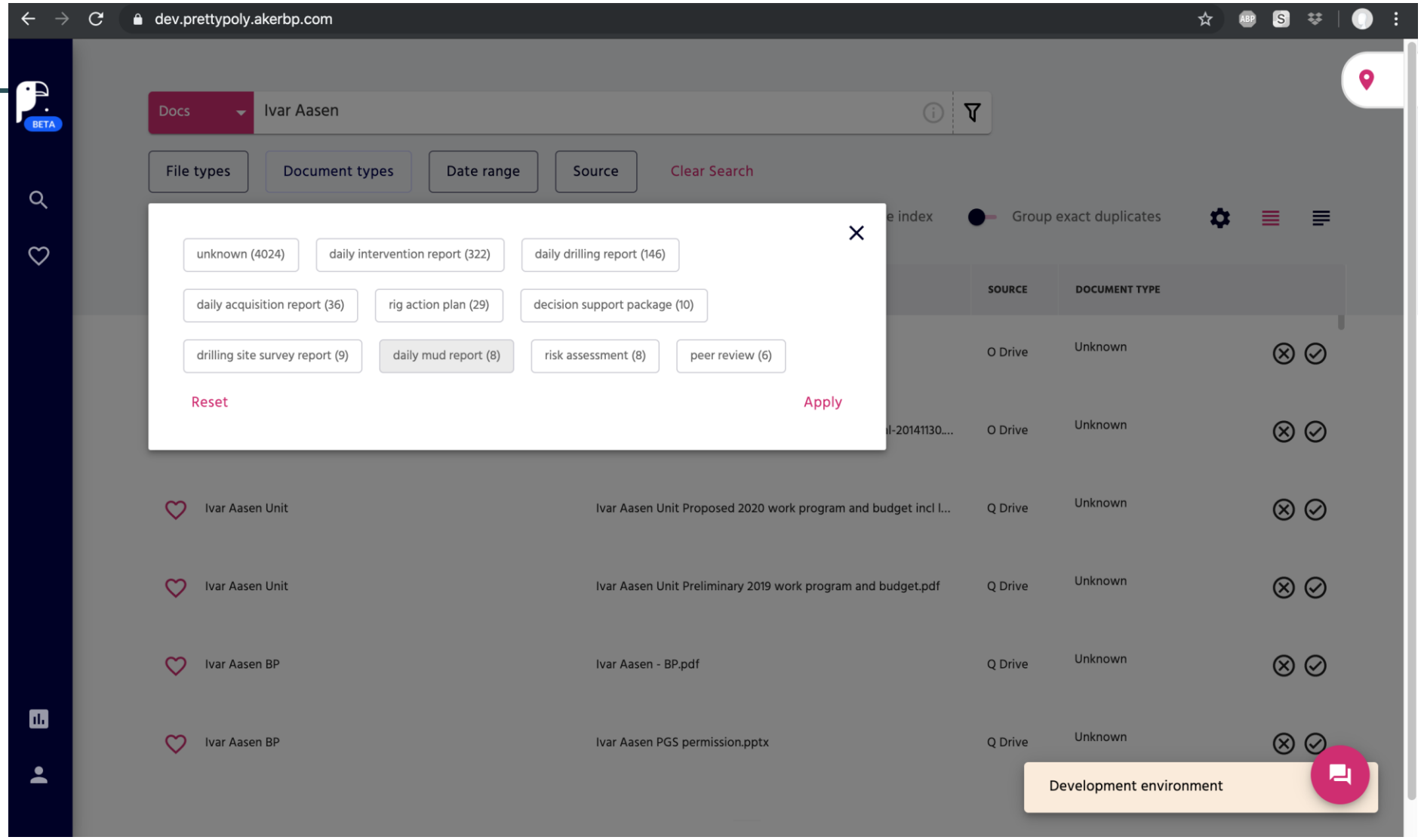

PrettyPoly
(Aker BP)
inmeta

# PrettyPoly

## Data search engine customised for oil and gas documents

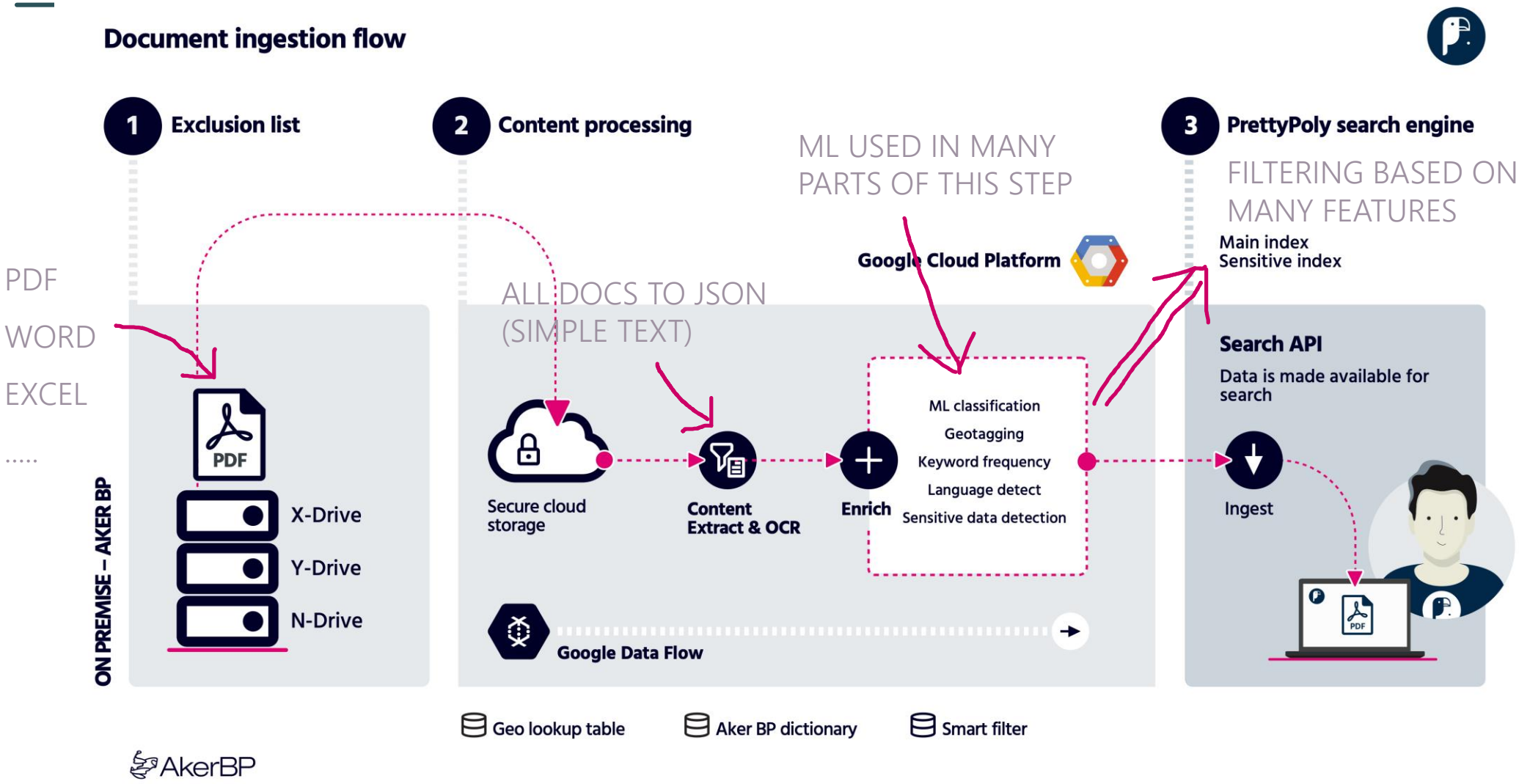
- Polygon search
- Geotagging
- Advanced query builder
- Collaboration/sharing
- Admin panel

- Document engine
- Sensitive content filtering
- Document tagging




inmeta

# PrettyPoly's document engine

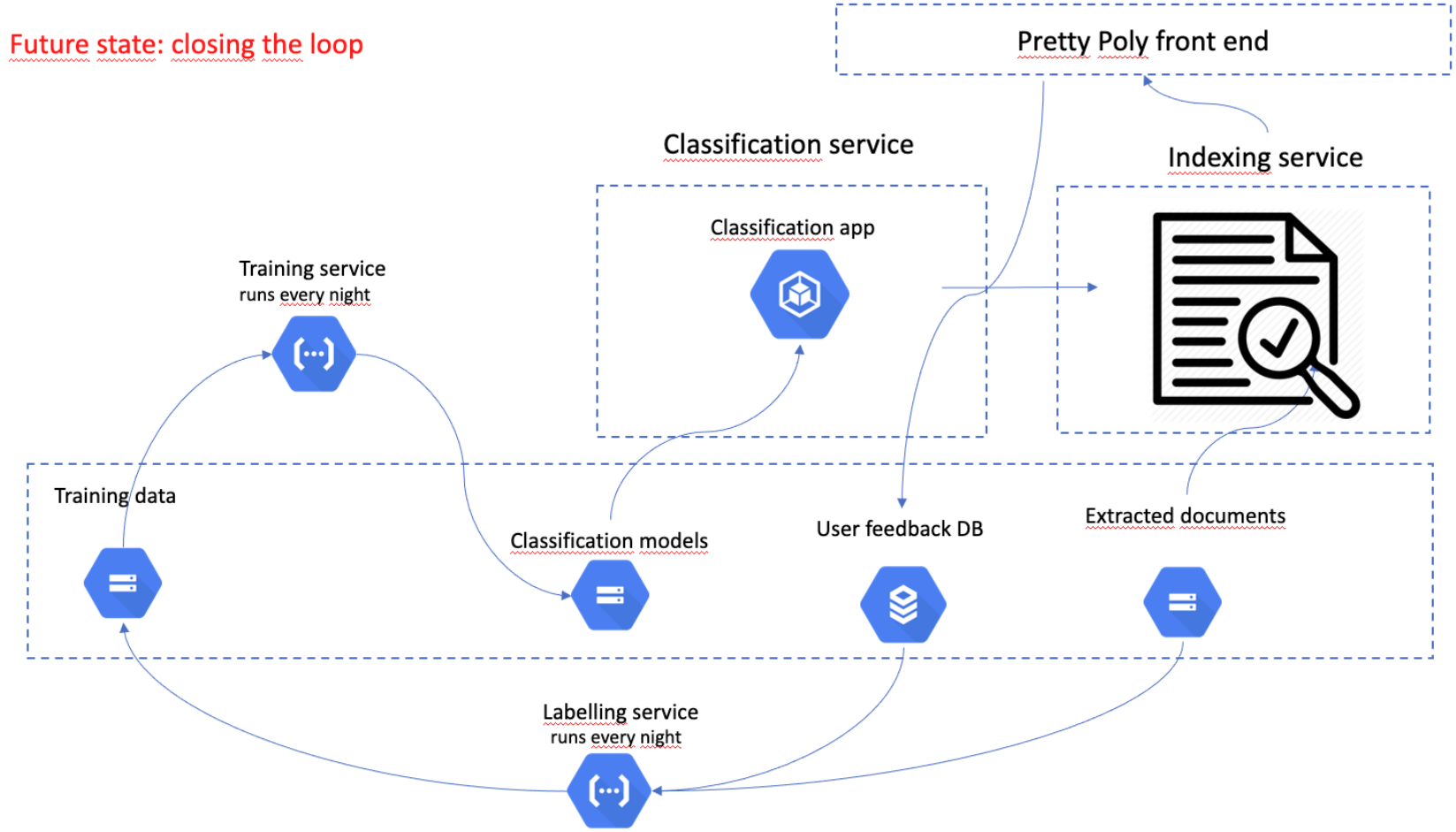



inmeta

# ML infrastructure for production

1. Single repo for whole project
   - Sub repos for model files and notebooks
2. Commits to master (with changes to ML) build a docker container
   - Grabs model files from model repo
3. Codebase has a python class per task (classification, keywords, language)
4. Containers stored in Google Container Registry
5. Containers then run (with auto scaling) on Google Kubernetes engine
6. Running containers host a *flask* app which exposes a classification endpoint
7. The dataflow ("outside of ML") handles most of the flow for us

# Training feedback loop




NOTE: As of now we manually retrain when we have new data/new model

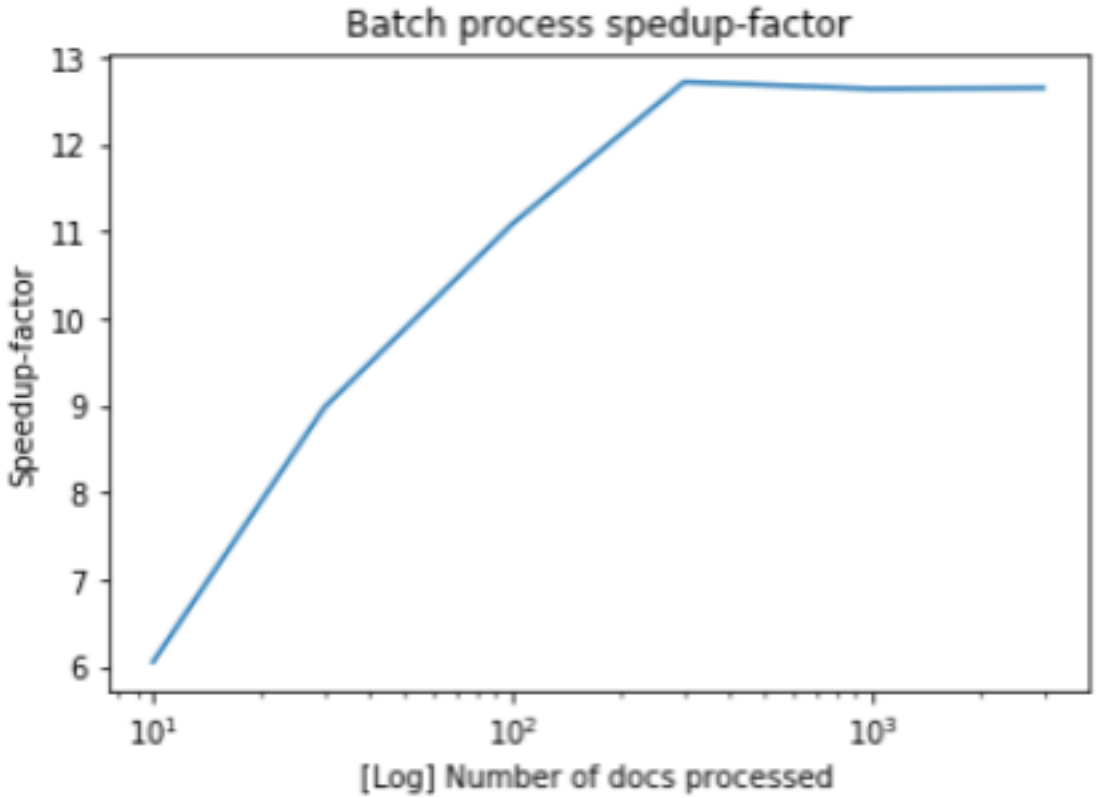
inmeta

# Training bucket

1. Single repo for whole project
   - Sub repos for model files and notebooks
2. Codebase has a python class per task (classification, keywords, language)
3. Commits to master (with changes to ML) build a docker container
   - Grabs model files from model repo
4. Containers stored in Google Container Registry
5. Containers then run (with auto scaling) on Google Kubernetes engine
6. Running containers host a *flask* app which exposes a classification endpoint
7. The dataflow ("outside of ML") handings taking the classification and taggings docs

# Notes and remarks

1. Data and feedback are crucial to success
2. Already a part of a google cloud project
3. Simple components
4. Kubernetes doesn't scale that well (scales down to 1, and scales up slowly)
5. Multi repos, and main repo requires the "models" repo is in line
6. Dataflow only allows single text prediction.
7. Have a large collection of notebooks that I run 'experiments' in (all prod code is python files)




inmeta


Summary
inmeta

# Summary

I don’t think there is a good solution in this space yet. Just try to get these things right

1. Data
2. Structure in your work habits
3. Tracking model performance, and ability to go back to old models
4. Software stuff
5. Limit complexity