ML in production

FunTech February 2019

m.andreev@conundrum.ai — Mark Andreev

Agenda

- About production
- Actuality of prediction
- From notebook to microservice
- Scale up your solution
- Monitoring & automatic problem solving
- Conclusion

https://clck.ru/FATUR

jupyter

docker

Main problems of production

Time

- Actuality of prediction

Data

- Inconstancy of data
- Difference between train / evaluation sets

Model

- Model sharing
- Model maintaining: regularly predict / re-train

24/7 without engineer

- Automatic monitoring
- Automatic problem solving

Actuality of prediction

Offline prediction (~3+ hour)

Churn prediction, User-Item recommendations

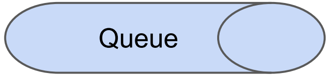
Actuality of prediction

Offline prediction (~3+ hour)

Churn prediction, User-Item recommendations

Online prediction (~5 minute)

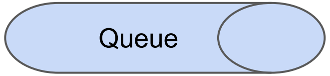
Classify photo, Rate announcement ads



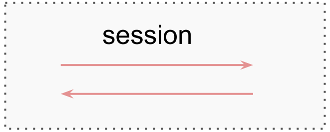
Actuality of prediction

Offline prediction (~3+ hour)
Churn prediction, User-Item recommendations

Online prediction (~5 minute)
Classify photo, Rate announcement ads



Realtime prediction (~300ms)
Search results, Ads recommendations
{Strong timeout SLA}



Inconstancy of data

Schema validation

Format validation using XML/Json schema

```
<!-- Schema Components -->
<xs:complexType name="baseComponent">
    <xs:complexContent> [29 lines]
</xs:complexType>
<xs:complexType name="componentWithFacets">
    <xs:complexContent>
        <xs:extension base="baseComponent"> [3 lines]
    </xs:complexContent>
</xs:complexType>
<xs:element name="schema">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="baseComponent">
                <xs:attribute name="type" use="required'
```

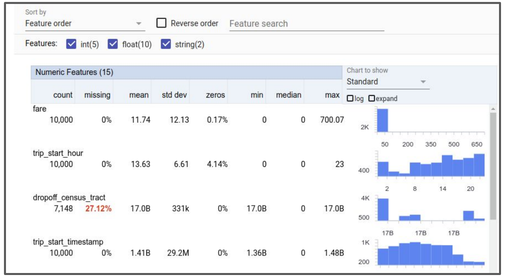
Inconstancy of data

Schema validation

Format validation using XML/Json schema

Data validation

Range validation. Test using hypotheses



Inconstancy of data

Schema validation

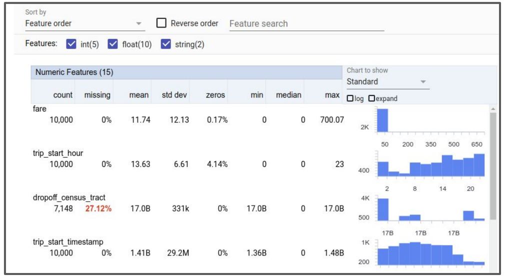
Format validation using XML/Json schema

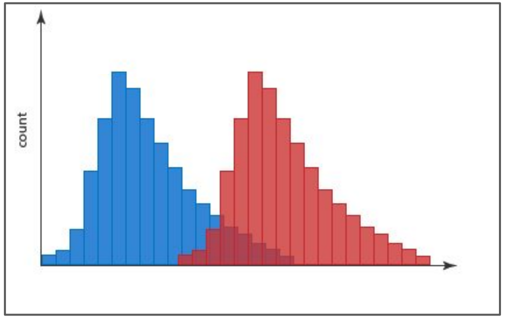
Data validation

Range validation. Test using hypotheses

Distribution validation

Descriptive statistics

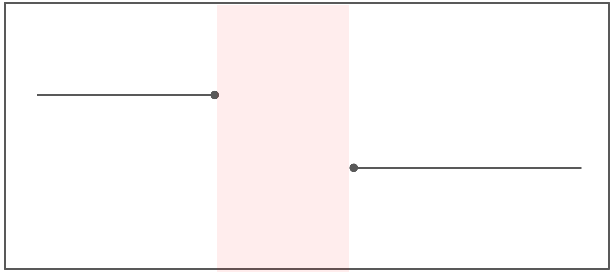




Difference between train / evaluation sets

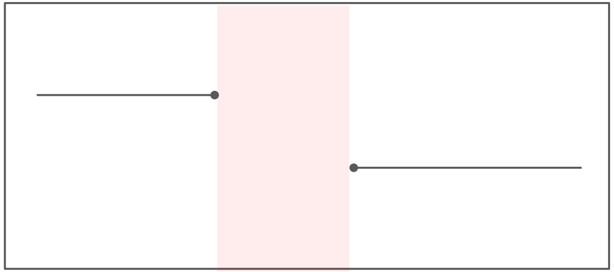
Train / Evaluation Time Gap

Time between train set and evaluation set

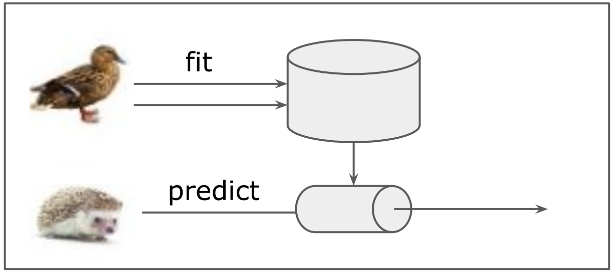


Difference between train / evaluation sets

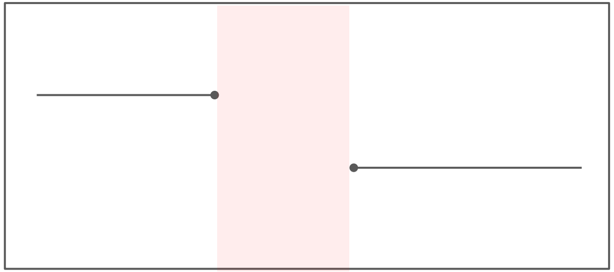
Train / Evaluation Time Gap
Time between train set and evaluation set



Feature extraction pipeline
Pipelines must be the same

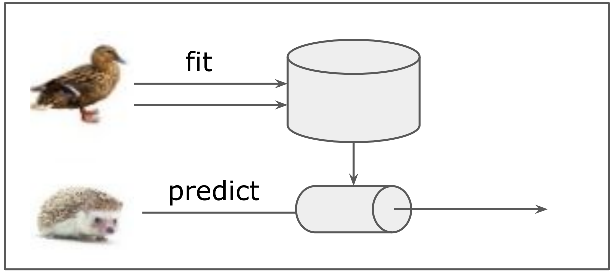


Difference between train / evaluation sets



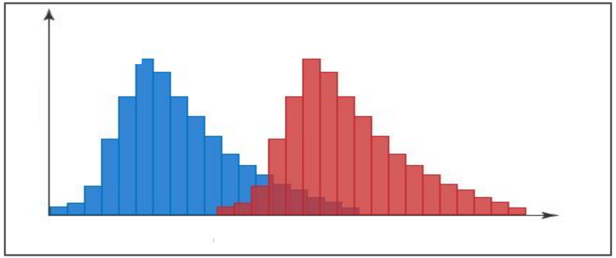
Train / Evaluation Time Gap

Time between train set and evaluation set



Feature extraction pipeline

Pipelines must be the same



Features distribution

Features distribution should be the same

How to share model

- solution.ipynb
- requirements.txt

- solution.py
- test_solution.py
- requirements.txt
- Dockerfile

Frozen dependencies

Python packages, System libraries

Tests

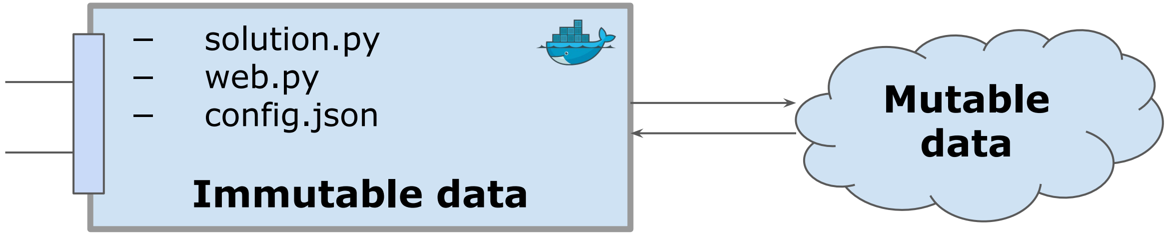
Unit tests, Integration tests, Exploration tests (hypothesis), Tests with data

Public interface

Expose your interface using REST (Flask, Tornado), describe it in Swagger

Stateless service

Stateless service



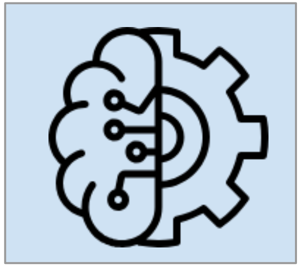
Extract state from service
Docker is an immutable container, extract the state outside

Public interface
Allow external connection only through public interfaces

Freeze service state
Save all dependencies and sub-dependencies

Scale up your service
Stateless allows us to linearly scale our solution

Scaling up using orchestration

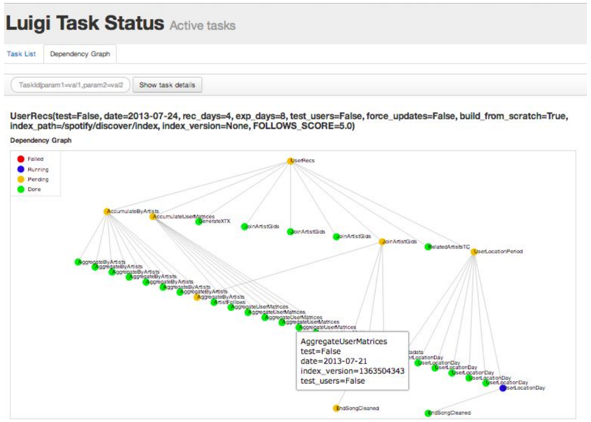


From pets to cattle

Regular offline prediction

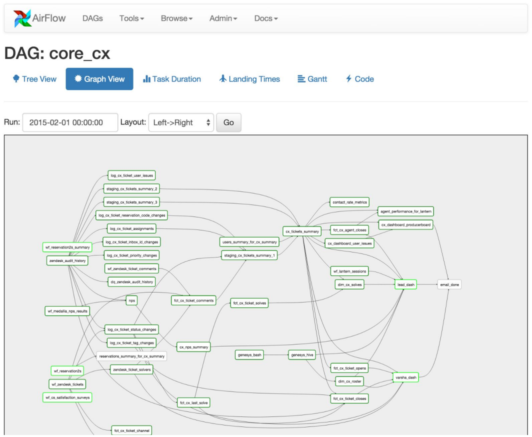
Luigi by Spotify

- Data pipeline framework
- More stable
- Scheduler is not included

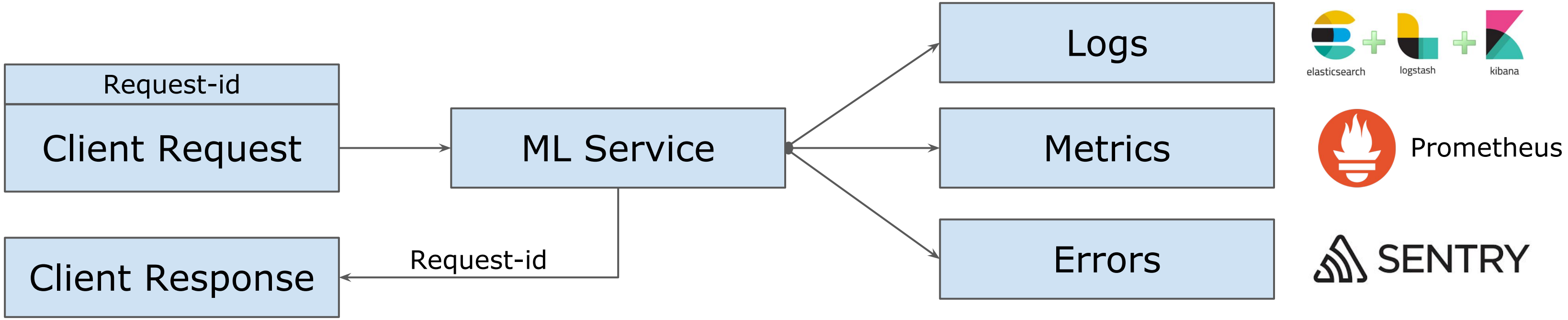


Airflow by airbnb

- Data pipeline framework
- More flexible
- More testable
- Pretty dashboard



Monitoring & automatic problem solving



Save your history

Use Logs, Metrics, Errors saving, Tracing for problem capturing and detection

Visualize your data through dashboards

Explicit is better than implicit. Visualize your key indicators

Graceful degradation.

Try to solve your problems automatically using spare models

Conclusion

- Check your inputs
- Containerize your solution
- Use Microservices Architecture
- Monitoring tools is your best friends
- Solve your problems automatically