Data Science at Booking.com

A few learnings

Lucas Bernardi

Big Data at Booking.com

Millions of properties (hotels, villas, cabins, etc)

Hundreds of millions of customers

Thousands of millions of transactions

Millions of Pictures

Millions of Reviews

Terabytes of new data every day

Data Science at Booking.com

Recommendations: Destinations, Hotels

Web site **Personalization**

Machine **Translation**

Automatic User **Image Tagging**

Causal Inference

Optimization

Anti-Pattern

Absence of evidence as proof of absence

Product Development at Booking.com

Come up with an idea

Create the least effort consuming Hypothesis that tests the idea

Implement it and run an experiment to test the Hypothesis

If the result is **Conclusive Positive**: Eat cake, create more hypothesis to test the same idea

If the result is **Conclusive Negative**: Something went wrong or the idea is just a bad one, analyse refine and iterate

Repeat

System Improvement at Booking.com

Come up with an idea

Implement it and run an experiment to check nothing is negatively impacted

If the result is **Conclusive Negative**: Bad news, roll-back, fix stuff, try again.

Otherwise: Good News, the change didn’t break anything.

Repeat

System Improvement at Booking.com

Come up with an idea

Implement it and run an experiment to check nothing is negatively impacted

If the result is **Conclusive Negative**: Bad news, roll-back, fix stuff, try again.

Otherwise: Good News, the change didn’t break anything.

Repeat

A/B Testing tests **B is different than A**

Hypothesis: CTR in version B is different than CTR in version A

Assume CTR in version B is the same as the CTR in version A

Compute the probability of the observed difference

If that probability is low then

- the assumption is considered wrong,
- the hypothesis is considered correct,
- we conclude that CTRs are different

Otherwise, the assumption cannot be deemed wrong: **No conclusion is made**

We want to test **B is not worse than A**

Running a standard A/B Experiment and failing is not proof of B being not worse than A

Absence of evidence is not proof of absence

Specific approaches must be used:

- Non-Inferiority Tests
- Lower Bounds of Confidence Intervals

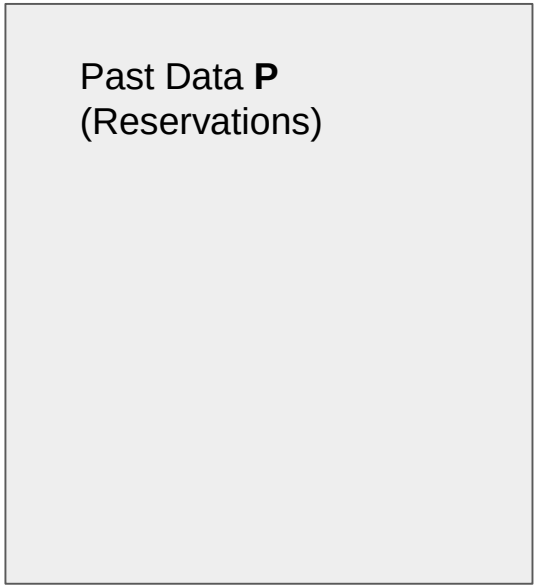
Challenge

The Graduation Syndrome

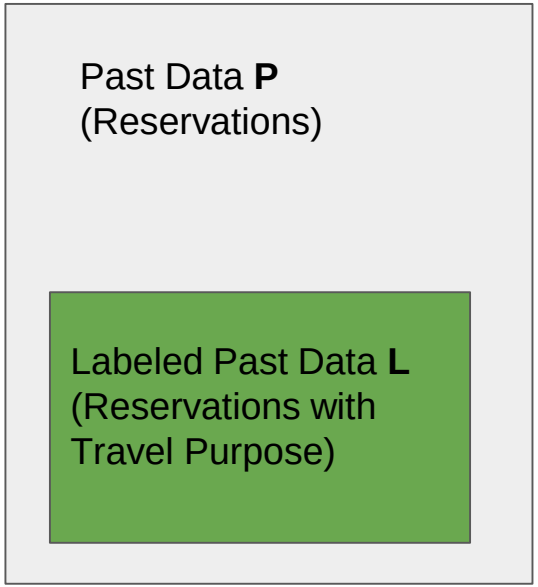
The Graduation Syndrome

An unhealthy obsession with the past and a refusal to live in the present. Symptoms include constant reminiscing about stories, people or events from college despite graduation having taken place in a different century than the one we now live in.

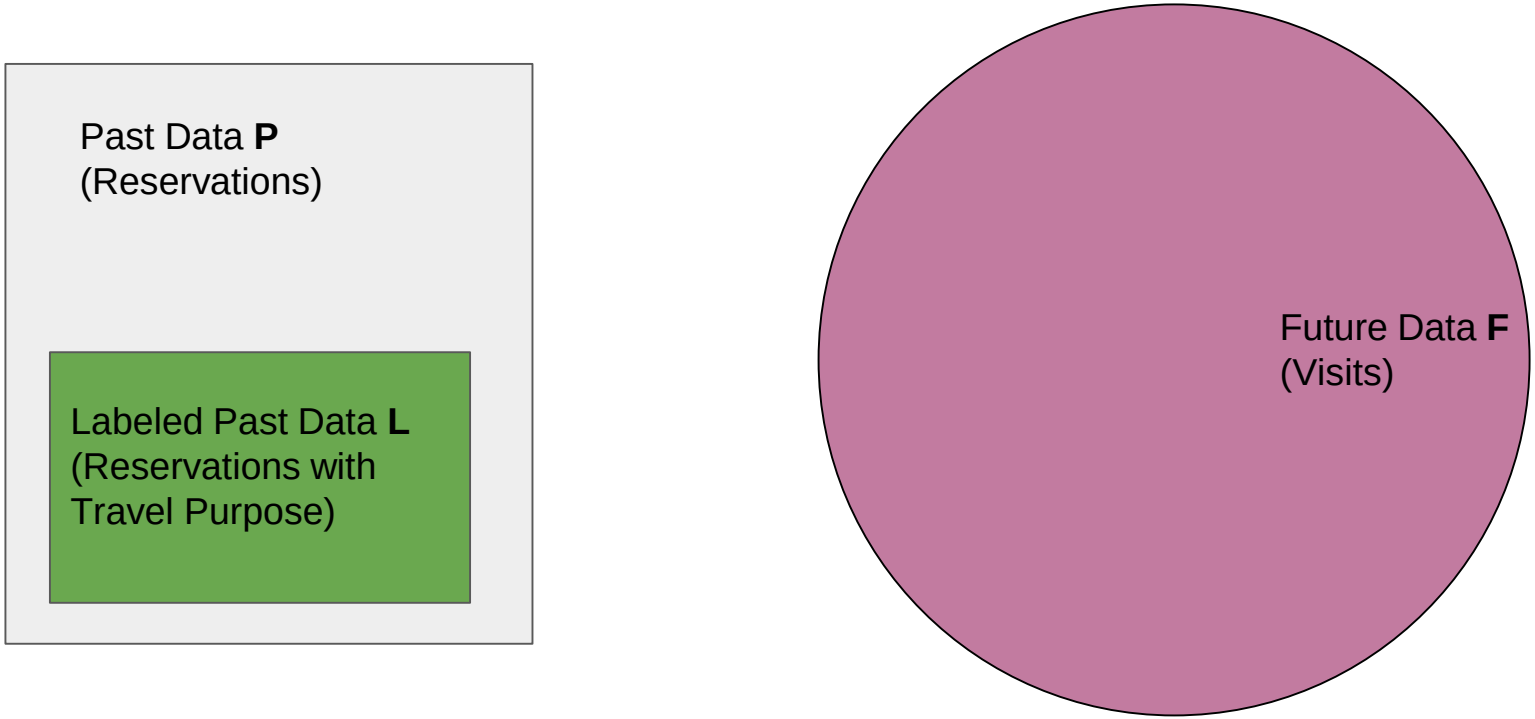
The Graduation Syndrome



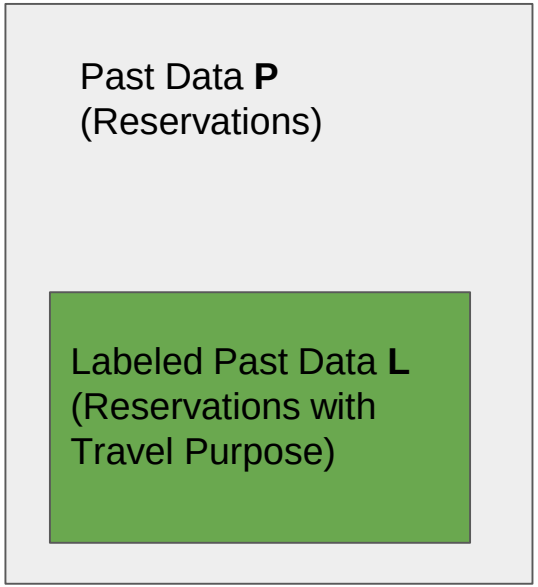
The Graduation Syndrome

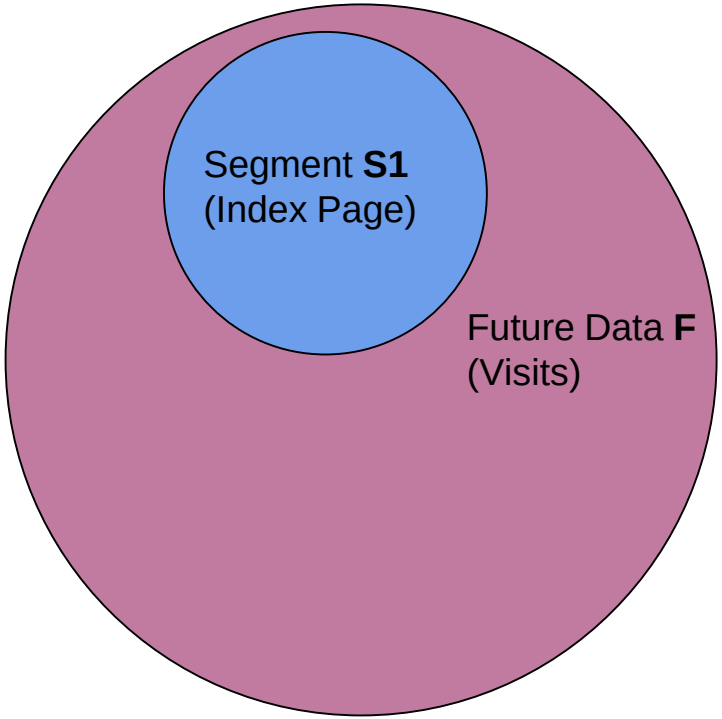


The Graduation Syndrome

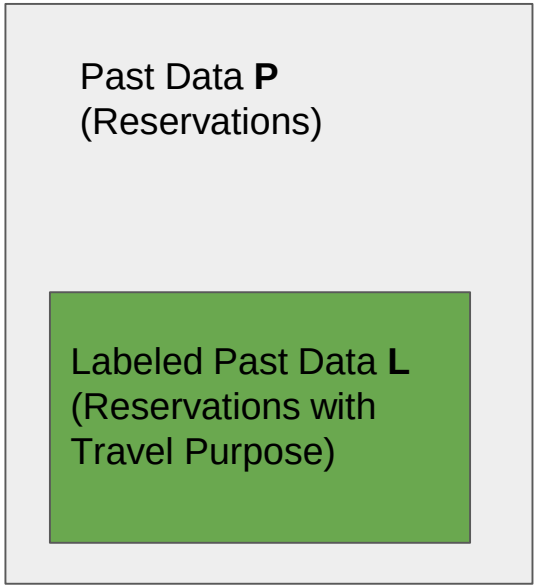


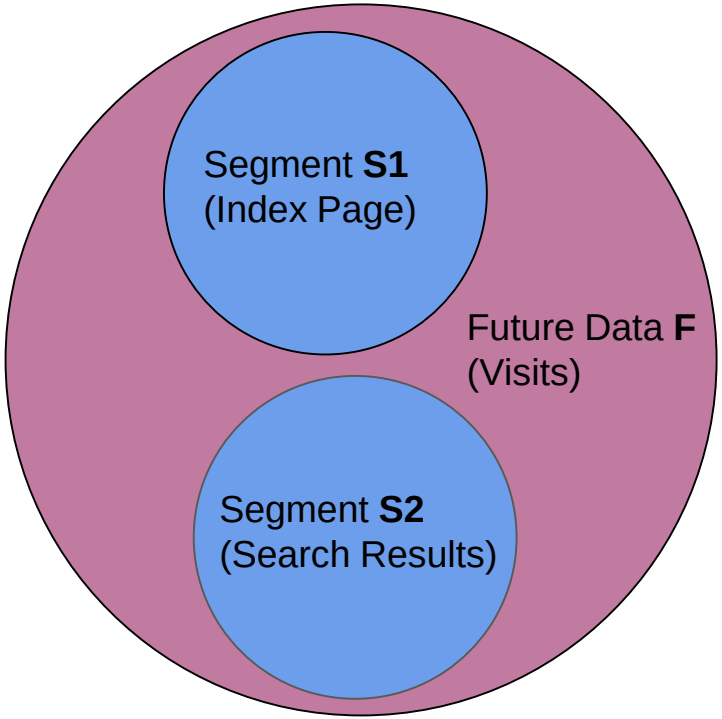
The Graduation Syndrome



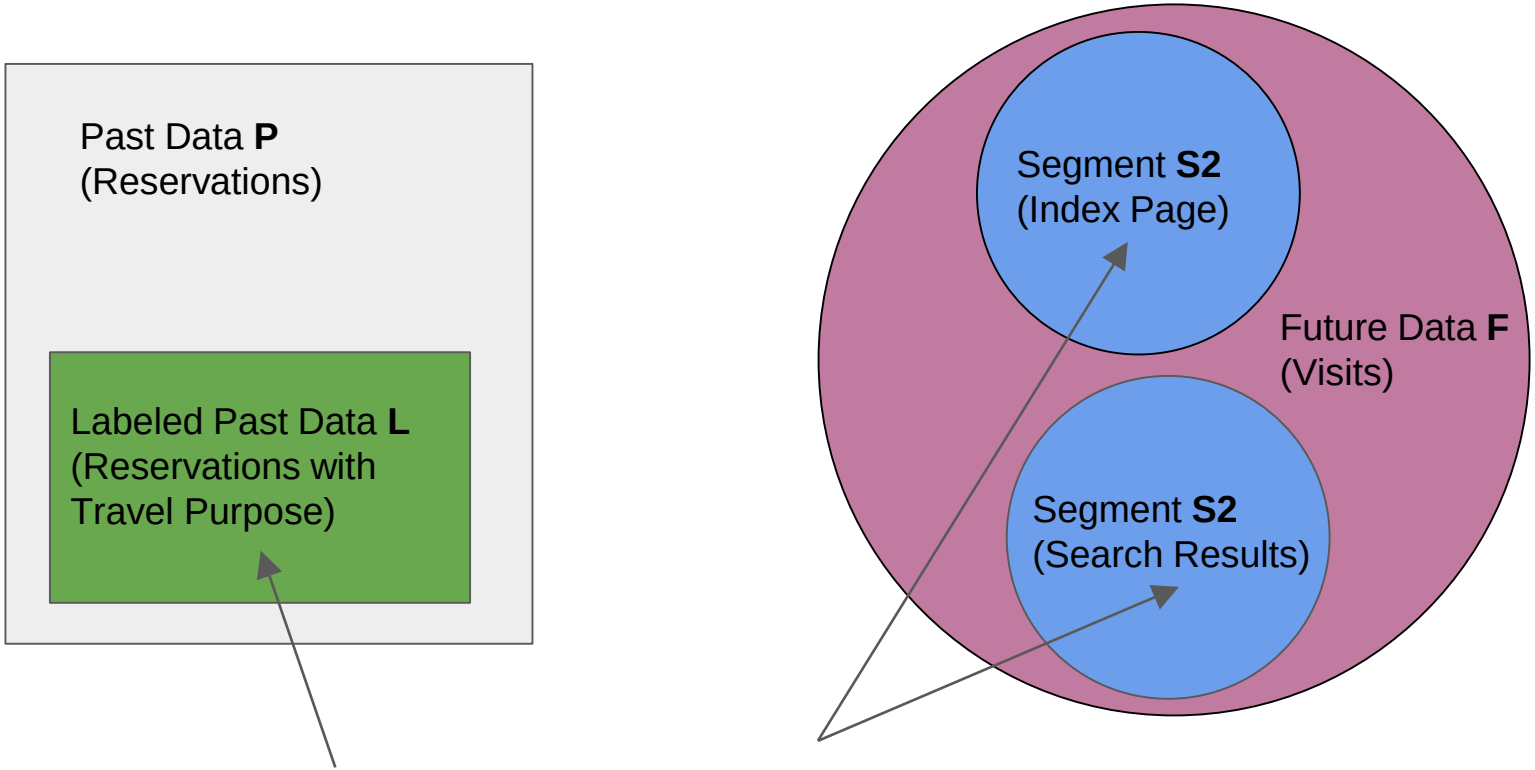


The Graduation Syndrome



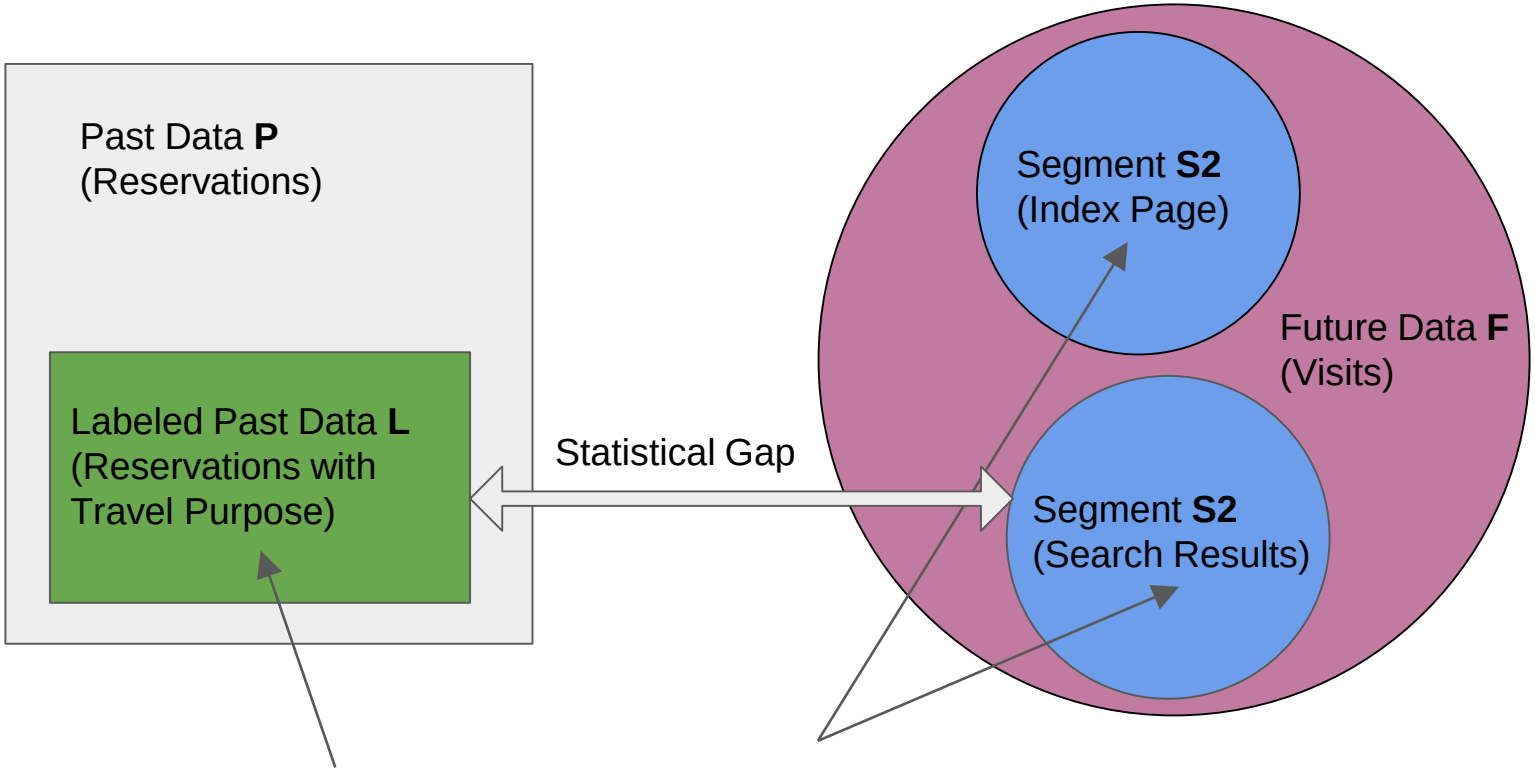


The Graduation Syndrome



Learn from **L** to predict on **S1** and **S2**

The Graduation Syndrome



Learn from **L** to predict on **S1** and **S2**

The Graduation Syndrome

Scenario 1

The learnt model is biased

If the bias is not big, the model is still valid

Scenario 2

The learnt model overfits L

The model is is invalid

The Graduation Syndrome

Scenario 1

The learnt model is biased

If the bias is not big, the model is still valid

Scenario 2

The learnt model overfits L

The model is is invalid

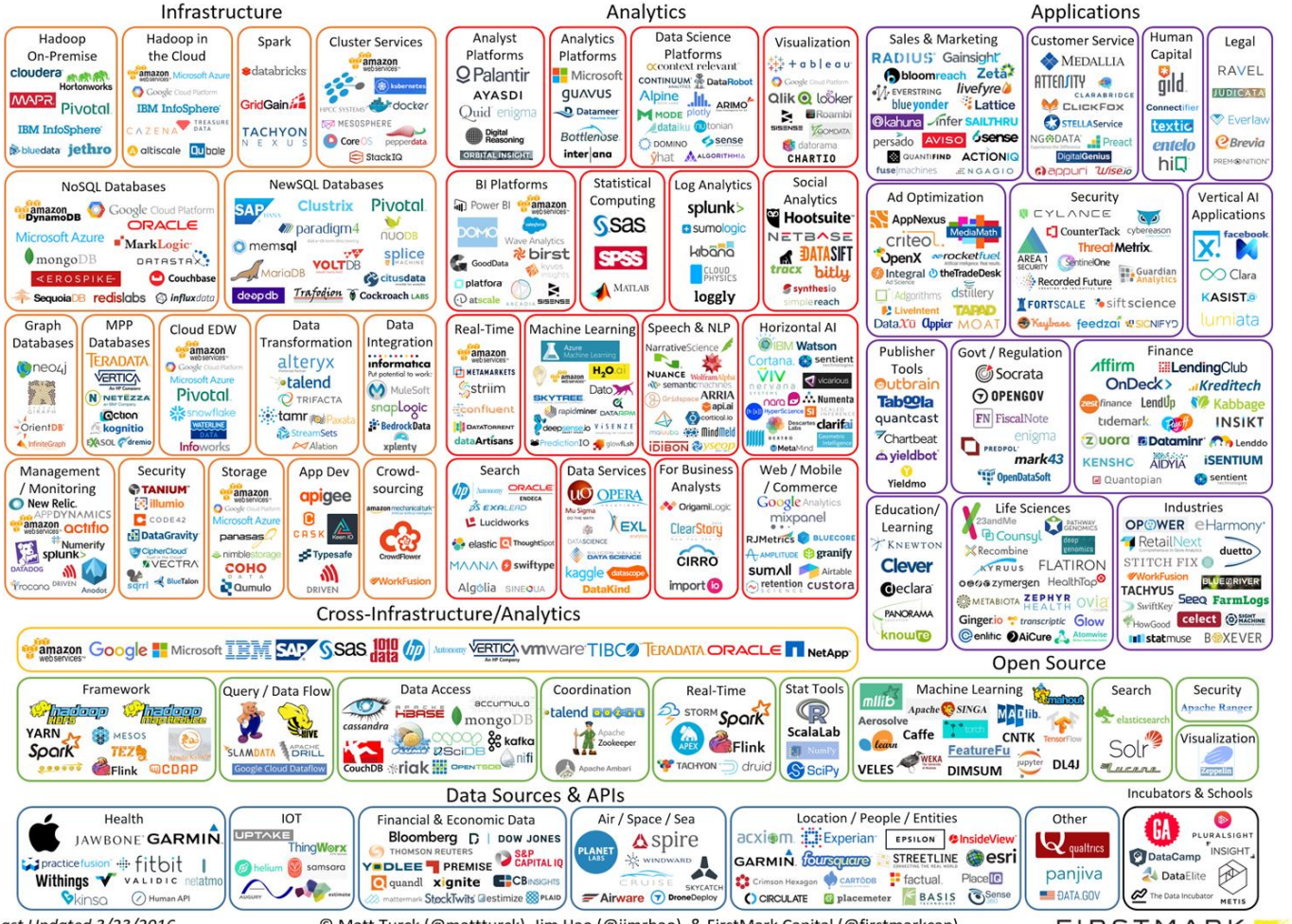
In General, **Label dependent metrics are not representative**

Cross Validation is not appropriate

Pattern

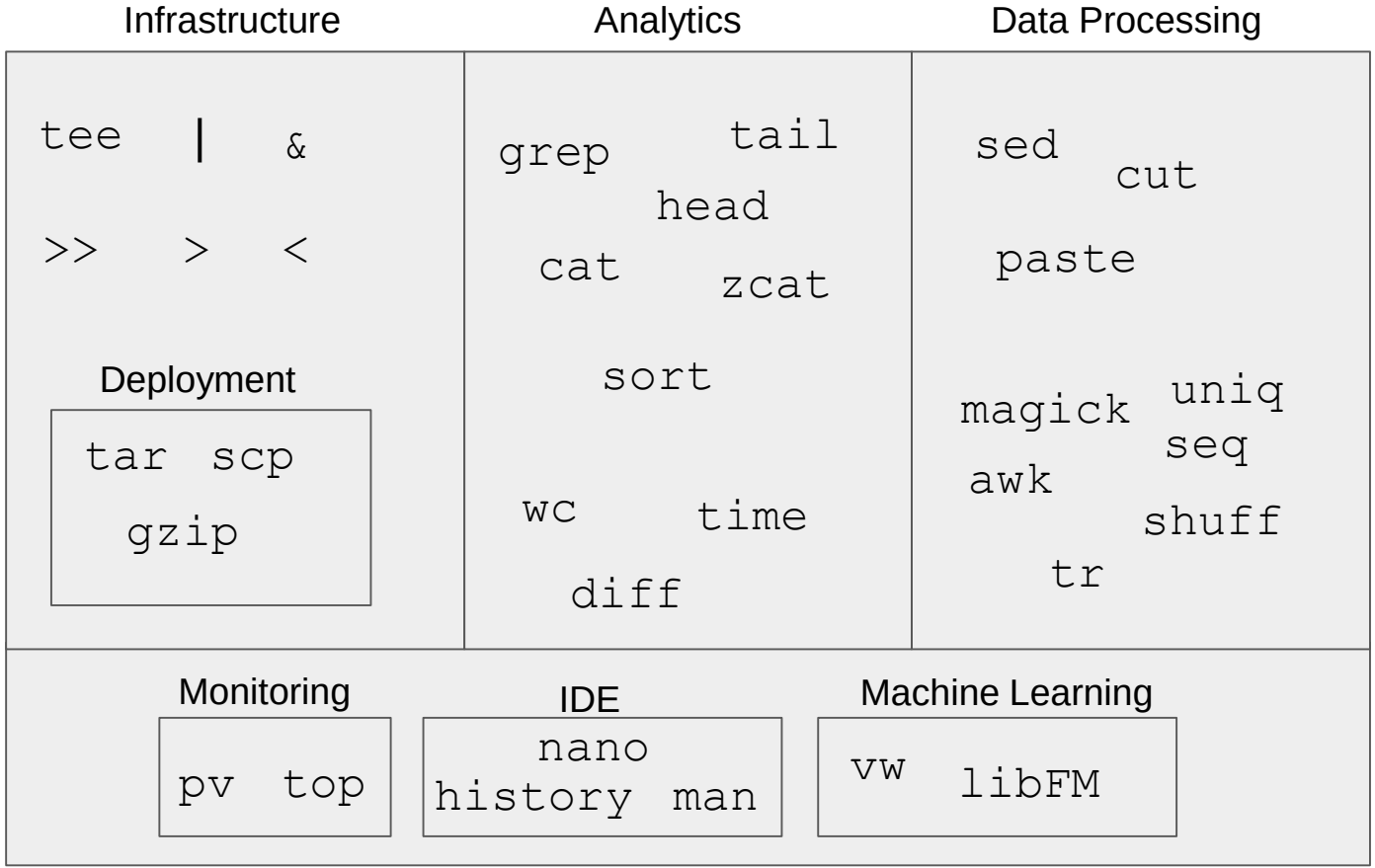
Out-of-core Pipelines

Big Data Landscape 2016 (Version 3.0)



Last Updated 3/23/2016

FIRSTMARK


Infrastructure
Analytics
Data Processing
tee | &
>> > <
Deployment
tar scp
gzip
grep tail
head
cat zcat
sort
wc time
diff
sed cut
paste
magick uniq
seq
awk
shuff
tr
Monitoring
pv top
IDE
nano
history man
Machine Learning
vw libFM

Out-of-core Learning: **Problem**

Given a large dataset stored in a data warehouse, build a Statistical Model using a pipeline that involves stages like:

1. fetching data
2. cleaning data
3. feature extraction
4. formatting data
5. model fitting
6. hyper-parameter search

Out-of-core Learning: **Structure**

Every stage of the pipeline is represented by a script

Scripts read data from standard input

Process data row by row

Output processed data to standard output

Use `|` and `tee` to communicate different stages

Use hard drive as a cache to communicate different pipes

Use out-of-core learning algorithms/implementations

Out-of-core Learning: **Example**

Train

```
hive -f get_training_data.hql | clean_data.py | feature_extraction.py |
csvvw.py | vw -b 29 -loss_function logistic -f mymodel.vw;
```

Predict

```
hive -f get_test_data.hql | clean_data.py | feature_extraction.py | csvvw.py |
tee >(vw  --loss_function logistic -i mymodel.vw -t --probabilities -p
preds.txt) | awk '{print $1}' > expected.txt;
```

Evaluate

```
compute_auc.py -truth expected.txt -preds preds.txt;
```

Out-of-core Learning: **Consequences**

Scalability: Memory usage is decoupled from the numbers of rows

Efficiency: Stages run as independent process, many CPUs are used

Separation of Concerns: Each stage is encapsulated in a script

Extensibility: High interoperability allows for new tasks and stages to be easily plugged in

We are Hiring!

lucas.bernardi@booking.com