Booking.com

Water you talking about?

Ivana Rebic & Dennis Bohle | Tel Aviv | 26.02.2019

Reusable Holdout Sets

Overfitting on the Kaggle public leaderboard.

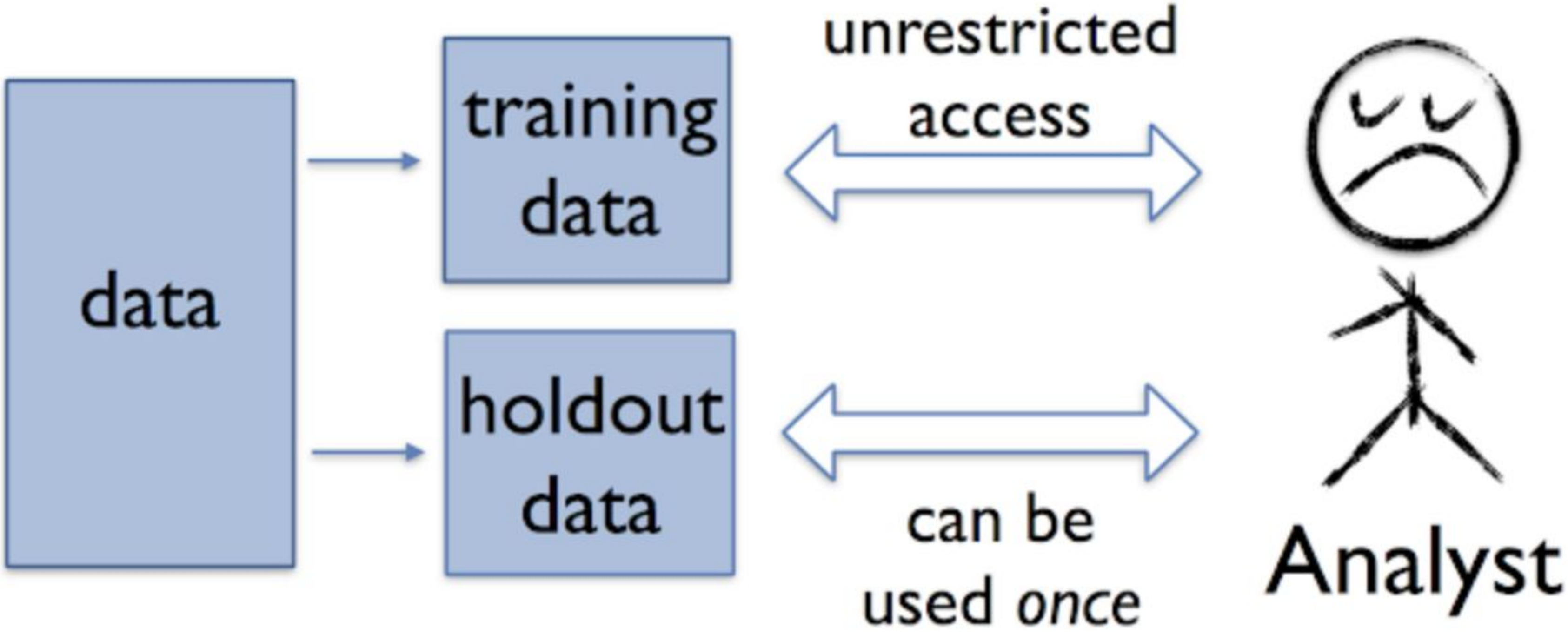
Kaggle Iceberg Classification Challenge

Overview Data Kernels Discussion Leaderboard Rules

#	Δpriv	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	David & Weimin			0.0801	118	5d
2	▼3	Kohei and Medrr			0.0852	114	4d
3	▼105	**Overfitted-Winner 3**			0.0867	78	4d
4	—	Mark Rippetoe witnesses			0.0877	106	5d
5	▲3	beluga			0.0880	24	4d
6	▲3	Evgeny Nekrasov			0.0893	66	4d
7	▼6	Pavel Pleskov			0.0921	44	5d
8	▼444	**Extreme Overfitted**			0.0927	13	9d
9	▲3	AzAkhtyamov			0.0929	60	4d
10	▼3165	**Extreme Overfitted**			0.0929	95	5d

https://medium.com/randomai/another-look-into-over-fitting-33e15b044a5e

Standard Holdout method.

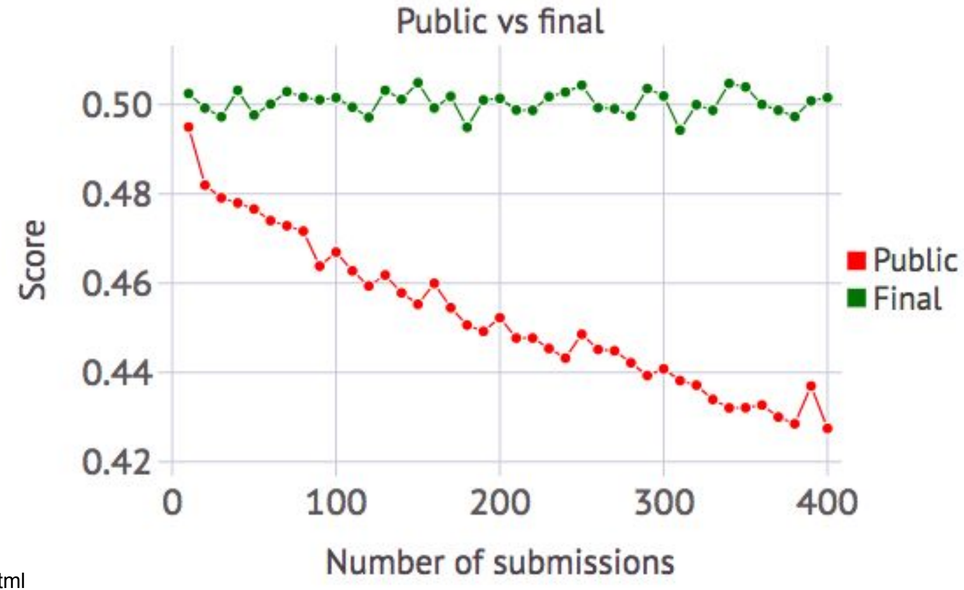


https://ai.googleblog.com/2015/08/the-reusable-holdout-preserving.html

Wacky Boosting Attack.

Algorithm (Wacky Boosting):

1. Choose $y_1, \dots, y_k \in \{0,1\}^N$ uniformly at random.
2. Let $I = \{i \in [k] \colon s_H(y_i) < 0.5\}$.
3. Output $\hat{y} = \text{majority}\{y_i \colon i \in I\}$, where the majority is component-wise.



http://blog.mrtz.org/2015/03/09/competition.html

The ladder algorithm.

Algorithm 1 Ladder mechanism

Input: Data set S, step size $\eta > 0$
Assign initial estimate $R_0 \leftarrow \infty$.
for round $t = 1, 2, \ldots$ **do**
  Receive function $f_t : X \to Y$
  if $R_S(f_t) < R_{t-1} - \eta$ **then**
    Assign $R_t \leftarrow [R_S(f_t)]_\eta$.
  else
    Assign $R_t \leftarrow R_{t-1}$.
  end if
end for

$$R_S(f) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)\,.$$

Avrim Blum, Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. CoRR, abs/1502.04585, 2015.

The ladder algorithm.

Definition 1 *Given an adaptively chosen sequence of classifiers* $f_1, \ldots, f_k$ *we define the* ***leaderboard error*** *of estimates* $R_1, \ldots, R_k$ *as*

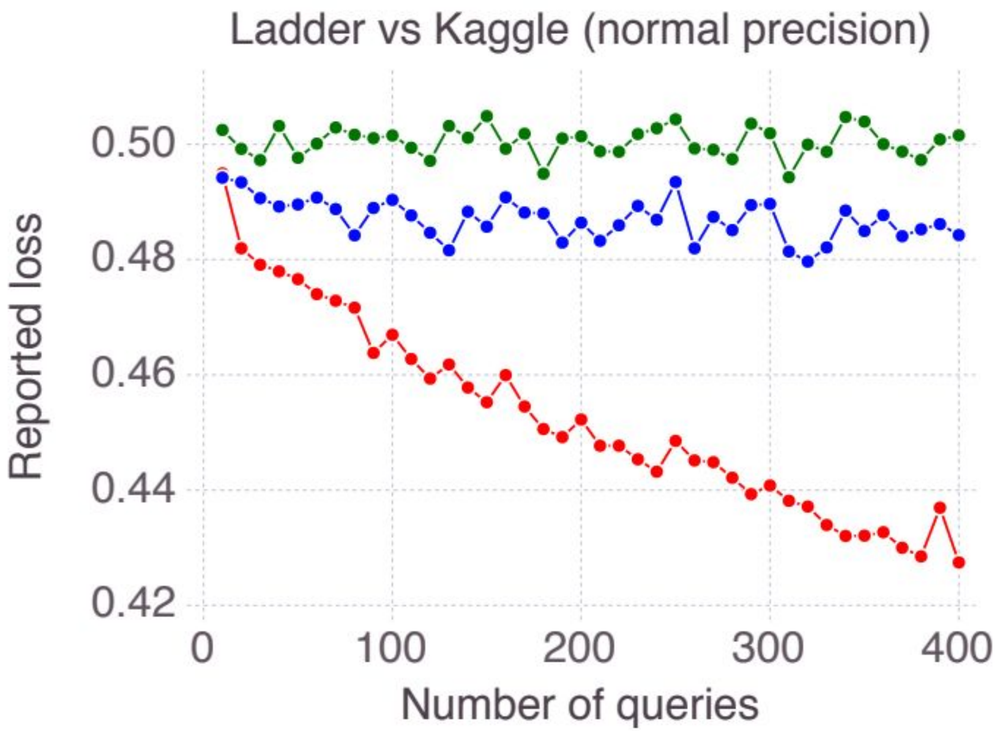
$$\mathrm{lberr}(R_1, \ldots, R_k) := \max_{1 \le t \le k} \left| \min_{1 \le i \le t} R_D(f_i) - R_t \right|$$

Theorem 1 *For any sequence of adaptively chosen classifiers* $f_1, \ldots, f_n$ *there exists* η *such that with high probability*

$$\mathrm{lberr}(R_1, \ldots, R_n) \le \mathcal{O}\left(\frac{\log^{1/3}(kn)}{n^{1/3}}\right).$$

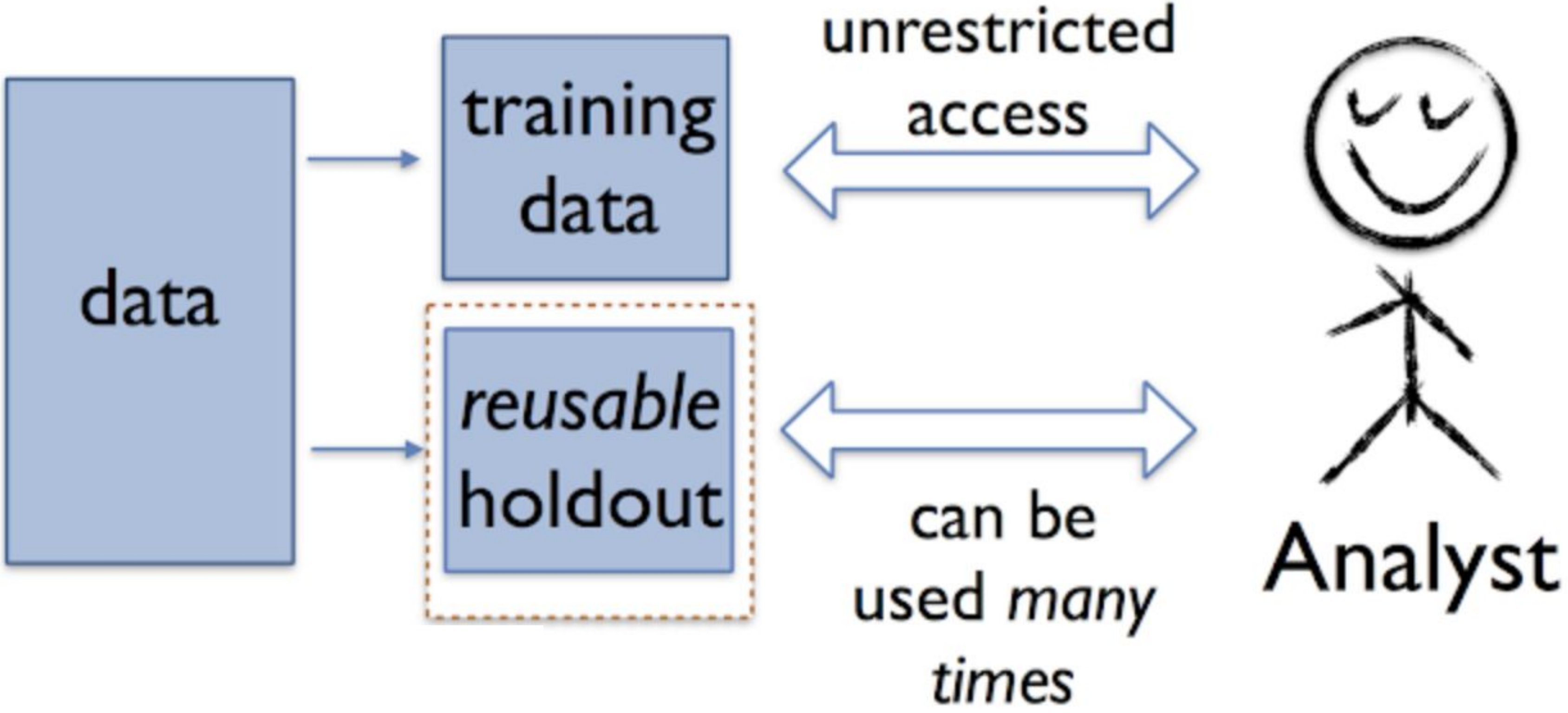
Avrim Blum, Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. CoRR, abs/1502.04585, 2015.

The ladder algorithm under wacky boosting.



Avrim Blum, Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. CoRR, abs/1502.04585, 2015.

Reusable Holdout Method.



https://ai.googleblog.com/2015/08/the-reusable-holdout-preserving.html

Feature Transformation Pipelines.

Booking.com

Model building flow.

Collect.	Transform.	Train.	Repeat.	Deploy.
Collect and clean the data	Apply the transformations to the data	Train a model	Repeat previous steps until happy	Deploy the model

Model generation at Booking.com.

Most of our data is stored in Hadoop. The most common tools for data extraction and munging are Spark and Hive, but it extends to Python, R and more.

Again for model building we use multiple tools.

Since many of the features are common across models, we have a central place for features that are supported and ready to plug in your model, called **Feature Store.**

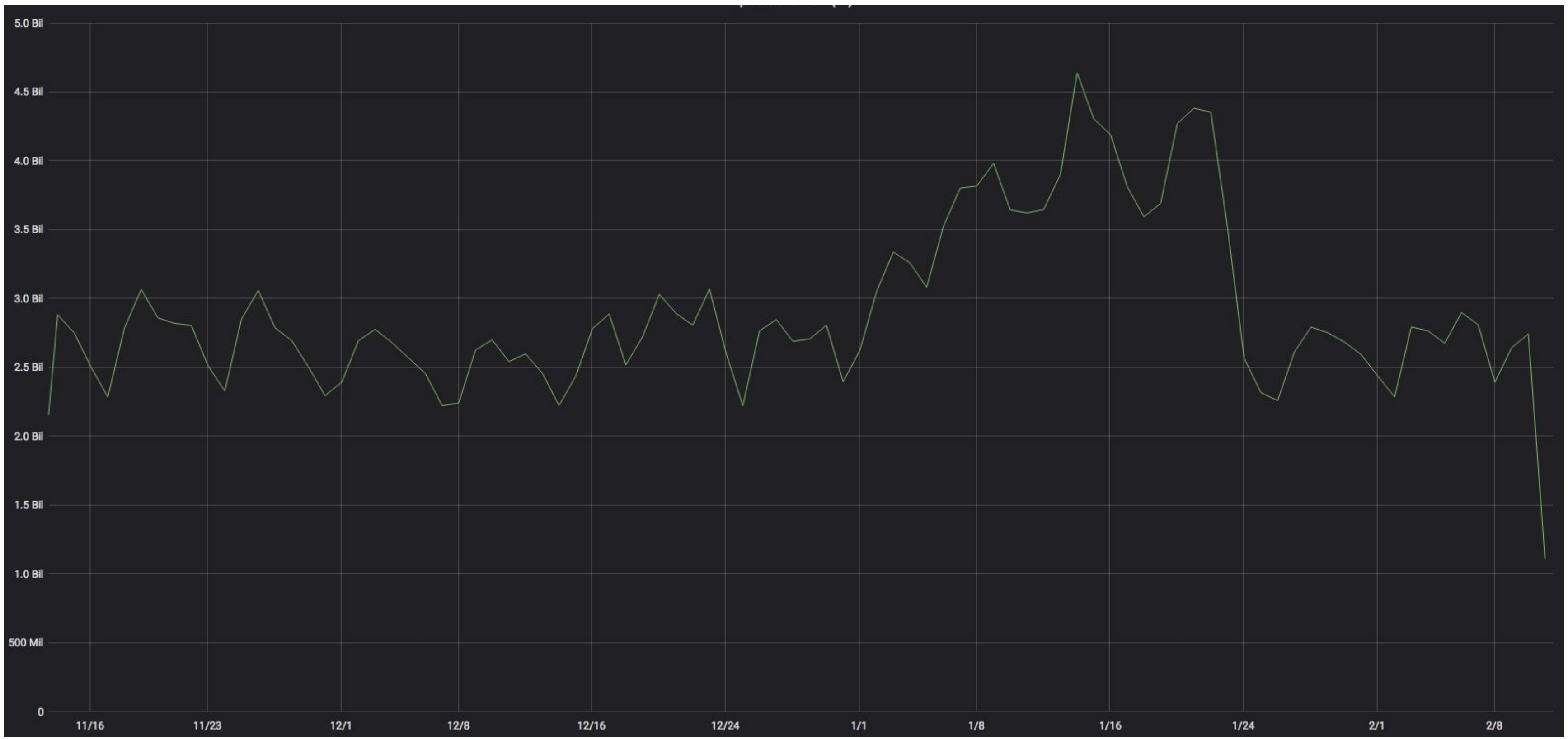
Custom feature transformation needed to be implemented separately when model was called in production.

Model serving at Booking.com .

We have model serving platform that provides tooling and monitoring to get model into production.

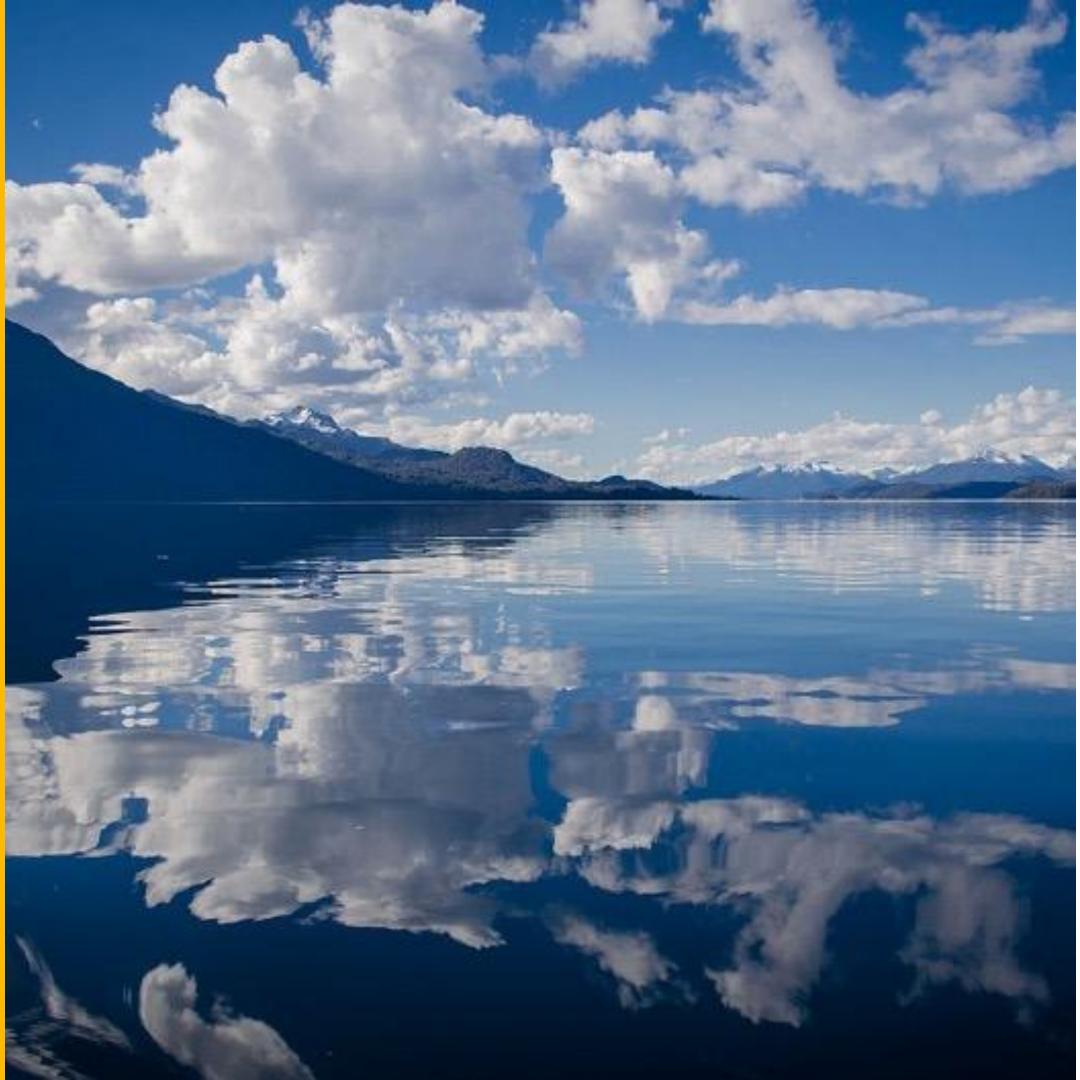
This is especially useful for models that will be used to predict on live data (for example, as a user searches on the website, or after a booking happens)

It is also a great place to discover existing models, where they are used and what experiments they have been part of.


5.0 Bil
4.5 Bil
4.0 Bil
3.5 Bil
3.0 Bil
2.5 Bil
2.0 Bil
1.5 Bil
1.0 Bil
500 Mil
0
11/16
11/23
12/1
12/8
12/16
12/24
1/1
1/8
1/16
1/24
2/1
2/8

Booking.com

How to make this faster and more reliable ?



Introducing Booking Transformations!

We implemented some of the often used transformations as Spark Transformers

Benefits:

- easily serializable
- can work on multiple columns simultaneously
- can be directly deployed
- less offline vs online disparity

Booking.com

Why not Spark MLLib Pipelines ?

Spark comes with the notion of transformers, estimators and pipelines

- A transformer transforms, . imputation with 0
- An estimator learns a transformer, imputation with the mean price
- A pipeline stitches multiple transformations together

What is there not to like about the transformers and estimators that come with spark?

- ➔ You can't easily deploy them (but you can store them)
- ➔ There is no way to extract the fitted dataframe, so you have to do double computation

Transformations.

Some of the available transformations:

- Imputation
  - {double, string} value imputation
  - {mean, majority} per some other category
- Bucketization
- Interactions
- Concatenations
- Scalers
  - Standardization
  - Normalization
- Target Estimator

Building pipeline.

Each transformation can be created as a separate object, and can then be fed into a transformation pipeline.

A pipeline can then be fit on a dataset to create a pipeline model.

This pipeline model can then be used to transform any dataset.

```
from booking.spark.fv.v1.transformations import Bucketizer, StandardScaler

bucketizer = Bucketizer()\
                .setInputCol("bookingWindow_Look")\
                .setOutputCol("bw_cat")\
                .setBuckets([0.0, 2.0, 5.0])\
                .setInclusionDirection("right")

scaler = StandardScaler()\
            .setInputCol("price")\
            .setOutputCol("price_scaled")

stages = [scaler, bucketizer]

pipeline = Pipeline().setStages(stages)

pipeline_model = pipeline.fit(dataset)

dataset_transformed = pipeline_model.transform(dataset)
```

Frame extractor.

The frame extractor is a simple identity estimator, whose function is to save a pointer to the final, transformed dataset at the end of a pipeline.

You use it by adding it to the end of your list of stages, and then use the extract() method to retrieve the transformed dataset

```
from booking.spark.fv.v1.transformations import FrameExtractor

frameExtractor = FrameExtractor()

stages = [scaler, bucketizer, frameExtractor]

pipeline = Pipeline().setStages(stages)

pipeline_model = pipeline.fit(dataset)

dataset_transformed = frameExtractor.extract()
```

Productionizing a pipeline.

The booking transformations are serializable into btl (bookings transformation language), which can be read by our model serving platform and used to create an online version of your transformation pipeline.

```
from booking.spark.fv.v1.transformations import PipelineWriter

path = '.'
writer = PipelineWriter(pipeline_model)
writer.write(path)
```

Benefits.

The booking transformations have a few benefits over the spark provided transformations

1. They do not rely on you creating vectors of your variables
2. They can be performed on multiple columns in parallel
3. They can be productionized using only two lines of code
4. They are lensuring less bugs and feature disparity
5. They shorten the time from model creation to model serving

Coding Target Encoding.

How to use and productionize target estimator.

Using bookings feature transformation pipeline makes it very simple to target encode any desired feature.
All what is needed to do is to specify and add Target Estimator object to the frame extractor!

```
targetEstimator = TargetEstimator()\
                      .setInputCol("var1")\
                      .setOutputCol("te_var1")\
                      .setFoldColumn("fold")\
                      .setSlope(10)\
                      .setInflectionPoint(50)
```

The inflection point determines how big a sample we still trust.
The slope determines how quickly this trust degrades, by defining the slope around the inflection point.

Note that these are hyper parameters of the encoding that should be tuned when using.

Fold column should already be be available in the dataset.
We will create a fold, and apply the transformation on our training dataset.

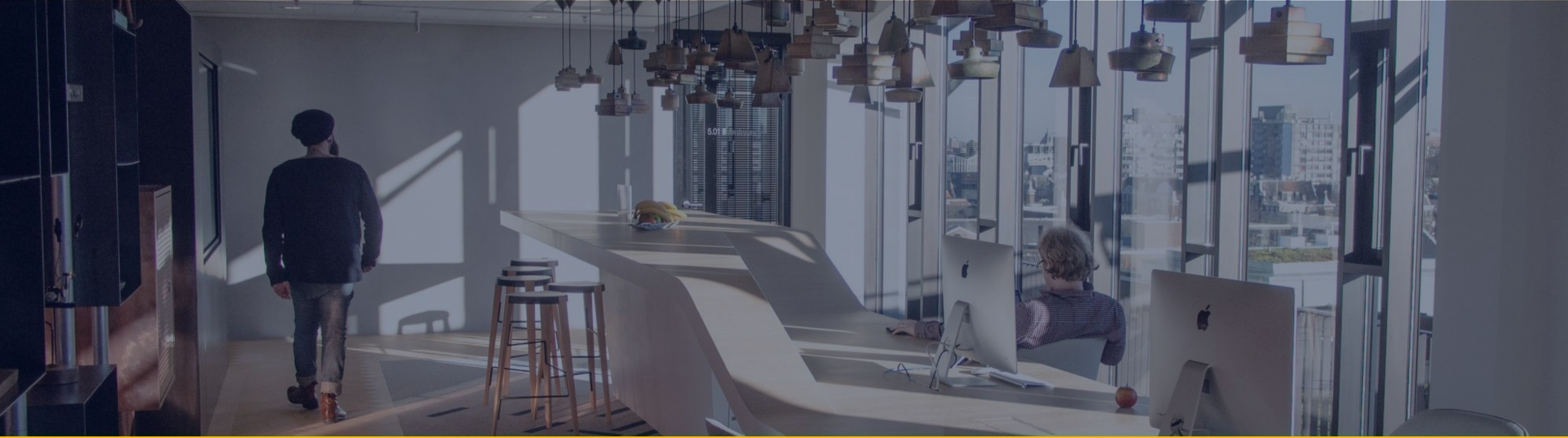
```
train = train.withColumn("fold", F.round(F.rand()*5))
targetEncoder = targetEstimator.fit(train)


train_te = targetEncoder.transform(train)
```

In order to also apply the target encoder to the test dataset, we have to tell the encoder not to use the folds (there is no danger of leakage in the test dataset)

```
paramList = []
paramList[targetEncoder.holdout] = False


test_te = targetEncoder.transform(train, paramList)
```

If we want to use target estimator, now we just need to add it to the list of our stages with other feature transformations and follow the same steps as mentioned before:

```
from booking.spark.fv.v1.transformations import FrameExtractor

frameExtractor = FrameExtractor()

stages = [scaler, bucketizer,targetEstimator frameExtractor]

pipeline = Pipeline().setStages(stages)

pipeline_model = pipeline.fit(dataset)

dataset_transformed = frameExtractor.extract()
```



Feature importance and explainability in tree models

Booking.com

Interpreting a complex model: Global vs Local.

Global interpretability

Relationship between the inputs and the dependent variable(s).

Local interpretability

Local interpretations promote understanding of small regions of the conditional distribution.

Game theory.

1. The sum of what everyone receives should equal the total reward
2. If two people contributed the same value, then they should receive the same amount from the reward
3. Someone who contributed no value should receive nothing
4. If the group plays two games, then an individual's reward from both games should equal their reward from their first game plus their reward from the second game

https://medium.com/@gabrieltseng/interpreting-complex-models-with-shap-values-1c187db6ec83

Additive feature attribution method.

Definition 1 Additive feature attribution methods *have an explanation model that is a linear function of binary variables:*

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{1}$$

where $z' \in \{0,1\}^M$, *M is the number of simplified input features, and* $\phi_i \in \mathbb{R}$.

Booking.com

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. pp. 4768–4777, 2017.

Local accuracy.

Local accuracy states that the sum of the feature attributions is equal to the output of the function we are seeking to explain.

Property 1 (Local accuracy)

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{5}$$

The explanation model $g(x')$ *matches the original model* $f(x)$ *when* $x = h_x(x')$, *where* $\phi_0 = f(h_x(\mathbf{0}))$ *represents the model output with all simplified inputs toggled off (i.e. missing).*

Booking.com

Lundberg SM, Erion GG, Lee SI (2018) Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Missingness.

Missingness states that features that are already missing are attributed no importance.

Property 2 (Missingness)

$$x'_i = 0 \implies \phi_i = 0$$

Missingness constrains features where $x'_i = 0$ *to have no attributed impact.*

Booking.com

Lundberg SM, Erion GG, Lee SI (2018) Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Consistency.

Consistency states that changing a model so a feature has a larger impact on the model will never decrease the attribution assigned to that feature.

Property 3 (Consistency) *Let* $f_x(z') = f(h_x(z'))$ *and* $z' \setminus i$ *denote setting* $z'_i = 0$. *For any two models* f *and* f', *if*

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \tag{7}$$

for all inputs $z' \in \{0,1\}^M$, *then* $\phi_i(f', x) \geq \phi_i(f, x)$.

Booking.com

Lundberg SM, Erion GG, Lee SI (2018) Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Shapely Values.

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right].$$

Booking.com

Lundberg SM, Erion GG, Lee SI (2018) Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Uniqueness.

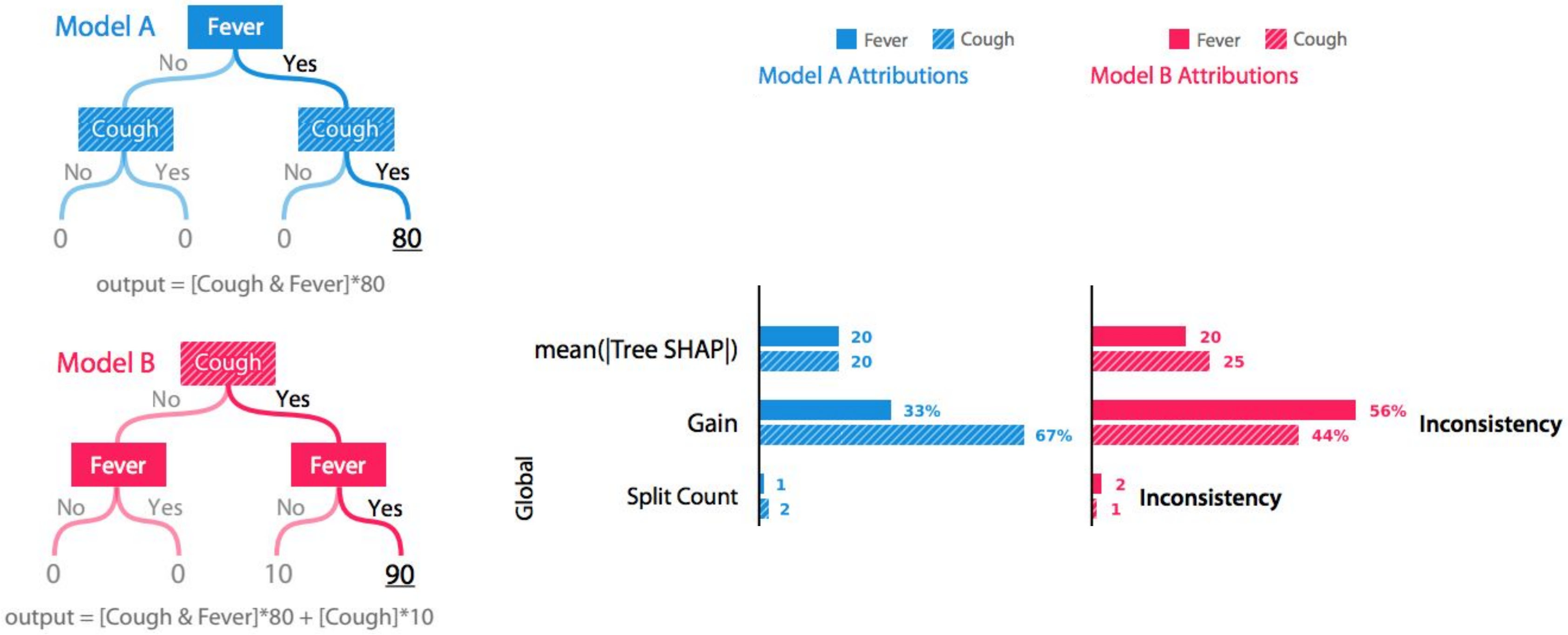
Theorem 1 *Only one possible explanation model g follows Definition 1 and satisfies Properties 1, 2, and 3:*

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)] \tag{8}$$

where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. pp. 4768–4777, 2017.

Inconsistencies in Gain and Split Count.

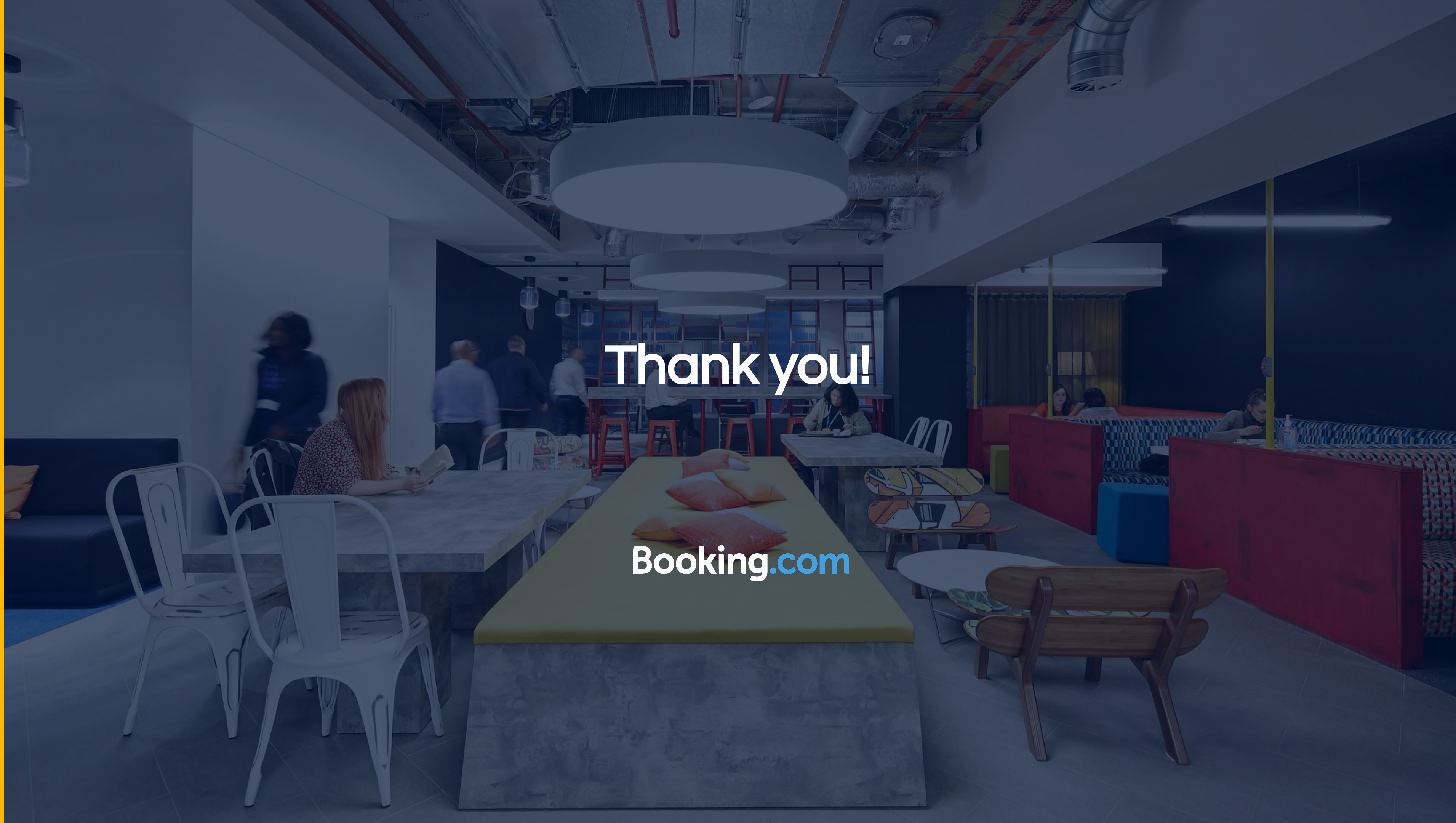


Lundberg SM, Erion GG, Lee SI (2018) Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Some open research questions at Booking.com

Just a few examples.

- Session based Information Retrieval
- Interference between A/B-Tests
- Testing under capacity constraints
- Debiasing feedback loops


Thank you!
Booking.com

Questions?