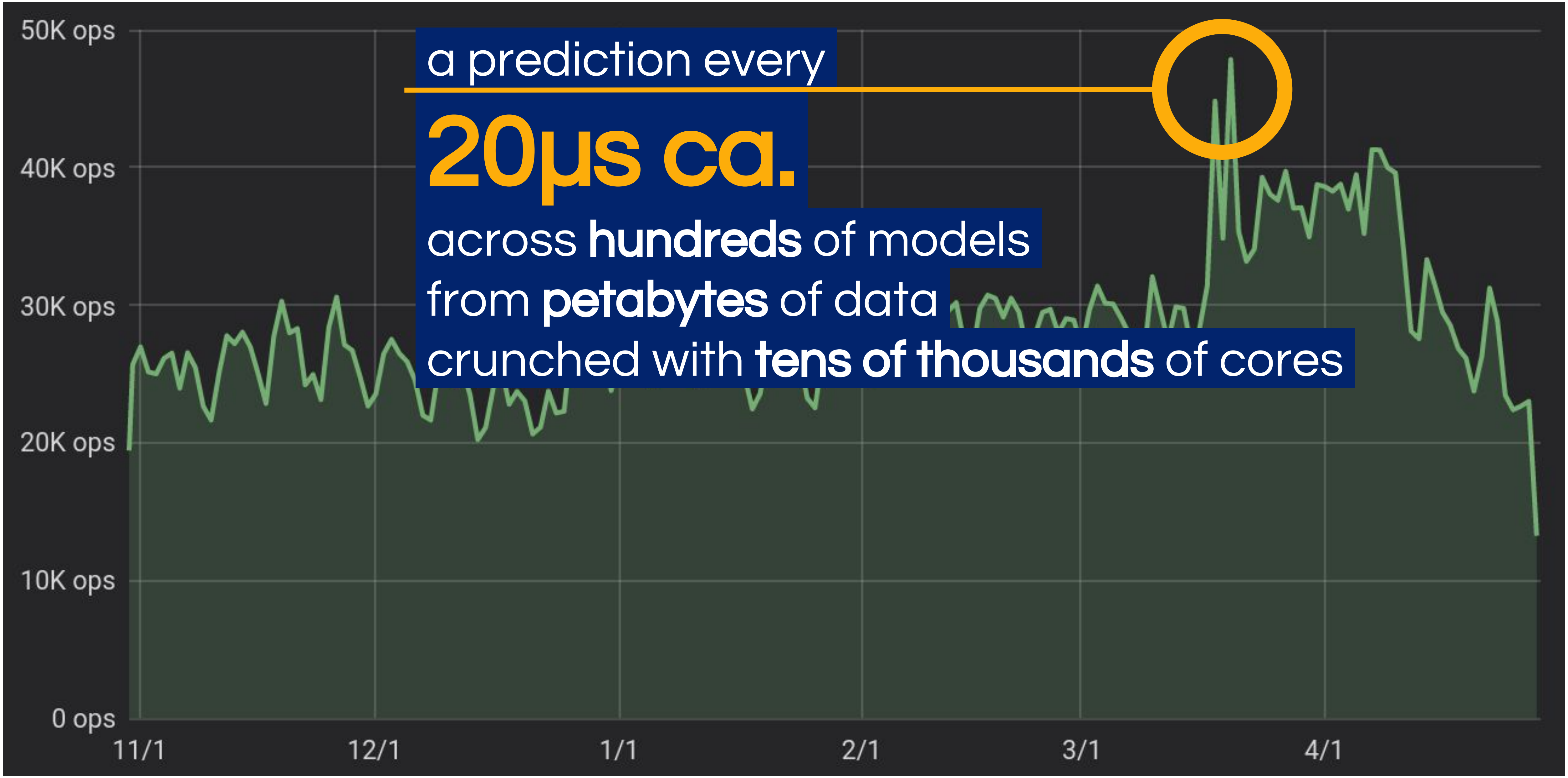
Booking.com

Water you talking about?
H2O Prague Meetup

Santi & Ivana Rebic | 16.05.2019


a prediction every
20µs ca.
across **hundreds** of models
from **petabytes** of data
crunched with **tens of thousands** of cores
50K ops
40K ops
30K ops
20K ops
10K ops
0 ops
11/1
12/1
1/1
2/1
3/1
4/1

Booking.com

Feature Transformation Pipelines.

Booking.com

Model building flow.

Collect.	Transform.	Train.	Repeat.	Deploy.
Discover, collect and engineer features	Apply transformations to the data	Train a model	Repeat previous steps until happy	Deploy the model

Model building flow.

Collect.	Transform.	Train.	Repeat.	Deploy.
Discover, collect and engineer features	Apply transformations to the data	Train a model	Repeat previous steps until happy	Deploy the model

Model building stack.

Collect.	Transform.	Train.	Repeat.	Deploy.
Proprietary	**(py)Spark**	**H2O**	**Jupyter**	**Proprietary**
Kafka	Pandas	VorpalWabbit	R Studio	Perl
Kafka Connect	HQL	Tensorflow	H2O	
	bash / Perl / C++ / ...	PyTorch	bash	

Model building stack.

Collect.	Transform.	Train.	Repeat.	Deploy.
Proprietary	**(py)Spark**	**H2O**	**Jupyter**	**Proprietary**
Kafka	Pandas	VorpalWabbit	R Studio	Perl
Kafka Connect	HQL	Tensorflow	H2O	MOJO?
	bash / Perl / C++ / ...	PyTorch	bash	

Model serving flow.

Serve.	Collect.	Transform.	Predict.	Profit!
Start serving a web request	Collect user data	Apply transformations to the data	Ask the model for a prediction	Use prediction to improve service

It’s a matter of parity.

Model training
(“the DS world”)

Collect. | Transform. | Predict.

Model serving
(“the dev world”)

Collect. | Transform. | Predict.

I’ll count reservations, but only those without cancellations

My buckets are [0, 5) and [5, 10]

Apply weights then biases to doubles

Collect. **Transform.** **Predict.**

Collect. **Transform.** **Predict.**

I’ll count reservations, with or without cancellations

My buckets are [-∞, 5] and (5, ∞]

Apply biases then weights to floats

Booking.com

I'll count reservations, but only those without cancellations

..was the reservation cancelled at instance time already?

Collect.

My buckets are [0, 5) and [5, 10]

what do you mean these can be null?

Transform.

Apply weights then biases to doubles

wait I rebuilt the mode and you should change thresholds

Predict.

Collect.

Transform.

Predict.

I'll count reservations, with or without cancellations

Expect some lag and sometimes I just won't

My buckets are [-∞, 5] and (5, ∞]

unless Stev clobbered the < operator AGAIN

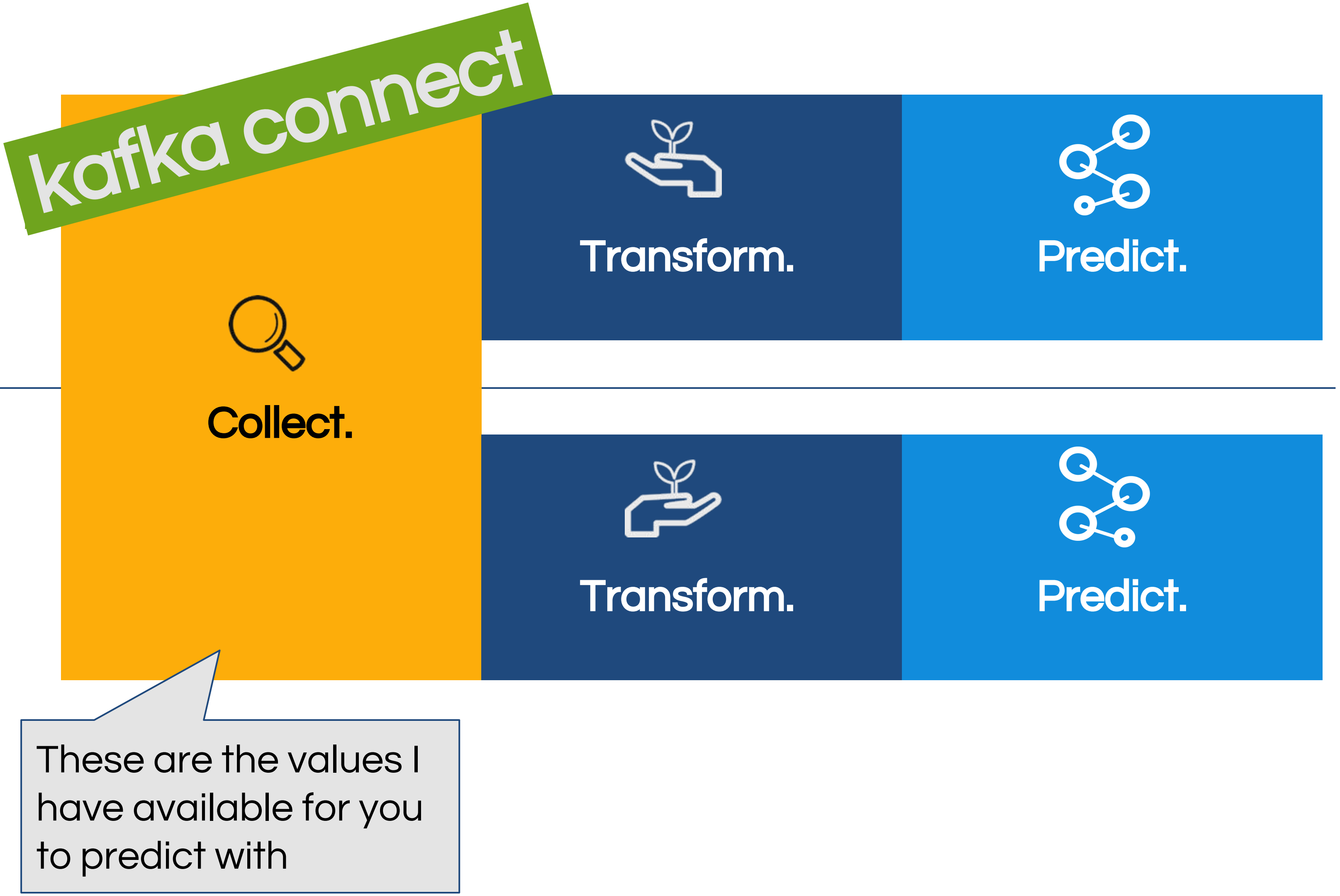
Apply biases then weights to floats

wait those numbers are unsigned right?

It's a matter of parity.

Model training

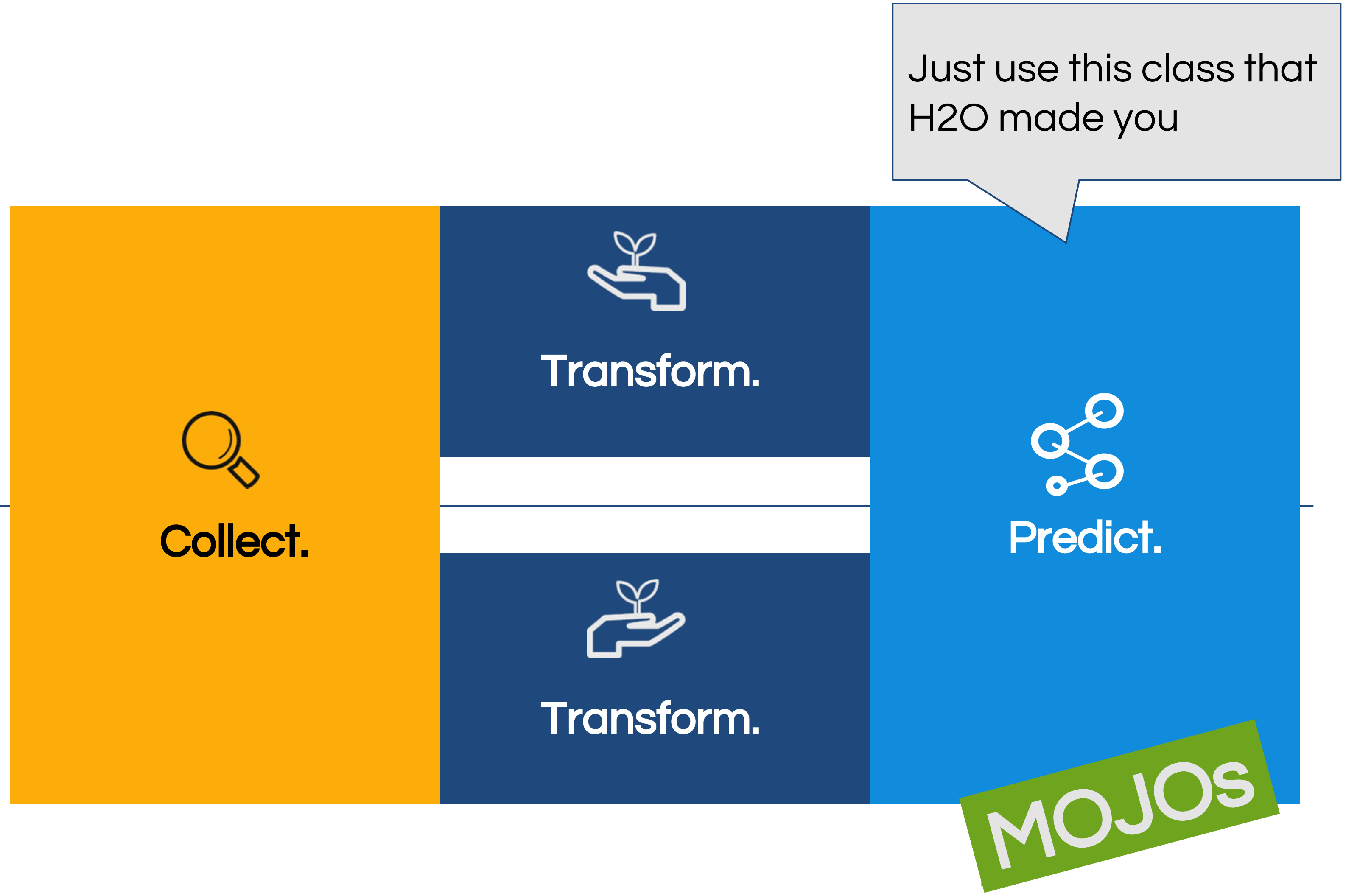
Model serving



It’s a matter of parity.

Model training

Model serving



It's a matter of parity.

I made you a spark transformation pipeline

Model training

Collect.

Transform.

Predict.

An exercise left to the reader

Model serving

Transform.

Great, but I can't use it

Booking.com

Spark MLLib Pipelines?

Spark comes with:

- **transformers**, which apply transformations (i.e. impute to zero)
- **estimators**, which produce transformers (i.e. impute to mean value)
- **pipelines**, which stitch multiple transformations together

Great! Except,

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

Introducing Booking Transformations!

We implemented some often used transformations as Spark Transformers

Benefits:

- easily serializable
- transforms multiple columns simultaneously
- built into our (proprietary) serving platform

```python
from pyspark.ml import Pipeline
from booking.spark.fv.v1.transformations import *

bucket_review_score = (
   Bucketizer()
   .setInputCol("review_score")
   .setOutputCol("review_cat")
   .setBuckets([6.0, 8.0, 9.0])
)

standardize_price = (
   StandardScaler()
   .setInputCol("price")
   .setOutputCol("price_scaled")
)

stages = [bucket_review_score, standardize_price]
# ...
```

no real numbers were used in the making of this slide.

```python
# ...



stages = [bucket_review_score, standardize_price]

pipeline = Pipeline().setStages(stages)

# pass 1: calculate standardization factors
pipeline_model = pipeline.fit(training)

# pass 2: apply factors to dataframe
training = pipeline_model.transform(training)
validation = pipeline_model.transform(validation)
```

```
# ...

get_frame = FrameExtractor()

stages = [bucket_review_score, standardize_price, get_frame]

pipeline = Pipeline().setStages(stages)

# pass 1+2: calculate and fit standardization factors
pipeline_model = pipeline.fit(training)

# just get the results
training = get_frame.extract()
validation = pipeline_model.transform(validation)
```

```
# ...

get_frame = FrameExtractor()

stages = [bucket_review_score, standardize_price, get_frame]

pipeline = Pipeline().setStages(stages)

# pass 1+2: calculate and fit standardization factors
pipeline_model = pipeline.fit(training)

# just get the results
training = get_frame.extract()
validation = pipeline_model.transform(validation)

PipelineWriter(pipeline_model).write('./model')
```

dumps learned values
in compressed JSON form

```
{"transformers": [
    {
        "jsonClass": "Bucketizer",
        "input": "review_score",
        "output": "review_cat",
        "buckets": [
            6.0,
            8.0,
            9.0
        ],
    },
    {
        "jsonClass": "Scaler",
        "offset": 4.333333333333333,
        "scale": 3.214550287658244,
        "input": "price",
        "output": "price_scaled",
    }
],
```

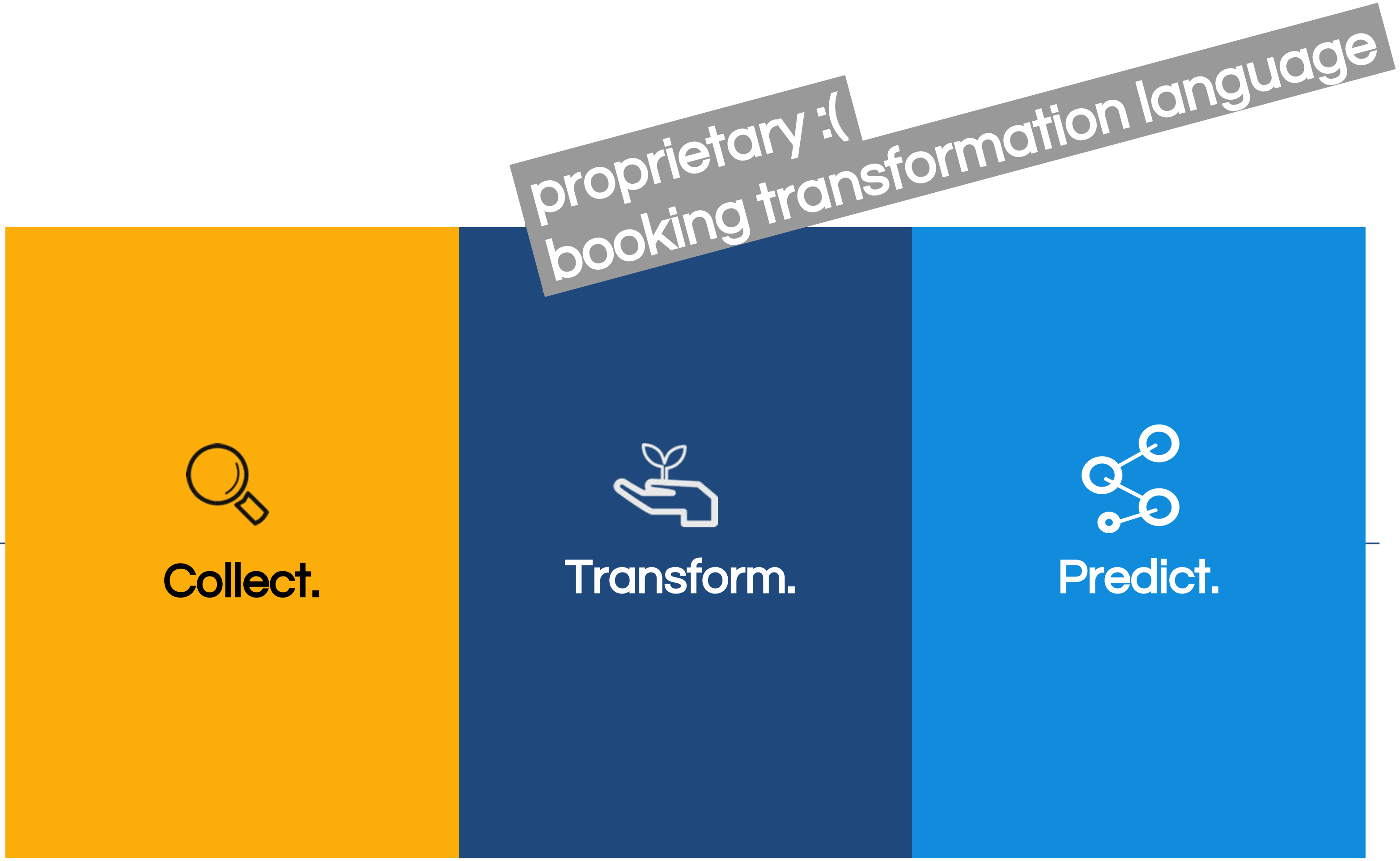
dumps learned values
in compressed JSON form

no real numbers were used in the making of this slide

Parity party!

Model training

Model serving



Transformations.

- Imputation (fixed or by category)
- Bucketization
- Interactions
- Concatenations
- Scaling
    - Standardization
    - Normalization
- **Target Estimator**

Coding Target Encoding.

Setup.

Encode categorical variables by looking at their relationship with the label.

In order to make the target encoder usable two problems need to be solved:

- prevent leakage
- deal with sparse variable values

Preventing leakage .

We'll divide the training dataset into a set of folds, and train $P_{f_i}(Y|X = X_i)$ for fold f_i by looking at the out of fold statistics.

We need to calculate

$$P(Y|X = X_i) = \frac{n_{iY}}{n_i}$$

We calculate these numerator and denominator values per partition of our dataset and for each fold.

This way we have to go through our dataset only once.

Dealing with the sparsity .

In order to deal with sparsity, we will smoothen the posterior probability $P_{f_i}(Y|X = X_i)$ with the prior probability $P_{f_i}(Y)$ in the following way:

$n_{i,j} = \sum_{k \notin f_j} Y_k$ is the number of positive examples out of fold

$$\lambda(n_{i,j})P_j(Y|X = X_i) + (1 - \lambda(n_{i,j})P_j(Y)$$

where we use the following lambda function:

$$\lambda(n) = \frac{1}{1 + \exp^{-\frac{(n-k)}{f}}}$$

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

Note that these are hyper parameters of the encoding that you should tune.

How to apply target estimator.

Using bookings feature transformation pipeline makes it very simple to target encode any desired feature.

Specifying and adding Target Estimator object to the frame extractor will do the trick!

```
targetEstimator = TargetEstimator()\
                .setInputCol("var1")\
                .setResponse("response")\
                .setOutputCol("var1_encoded")\
                .setFoldColumn("fold")\
                .setSlope(10)\
                .setInflectionPoint(50)
```

Fold column should already be be available in the dataset. So we will create a 5 way fold, and apply the transformation on our training dataset.

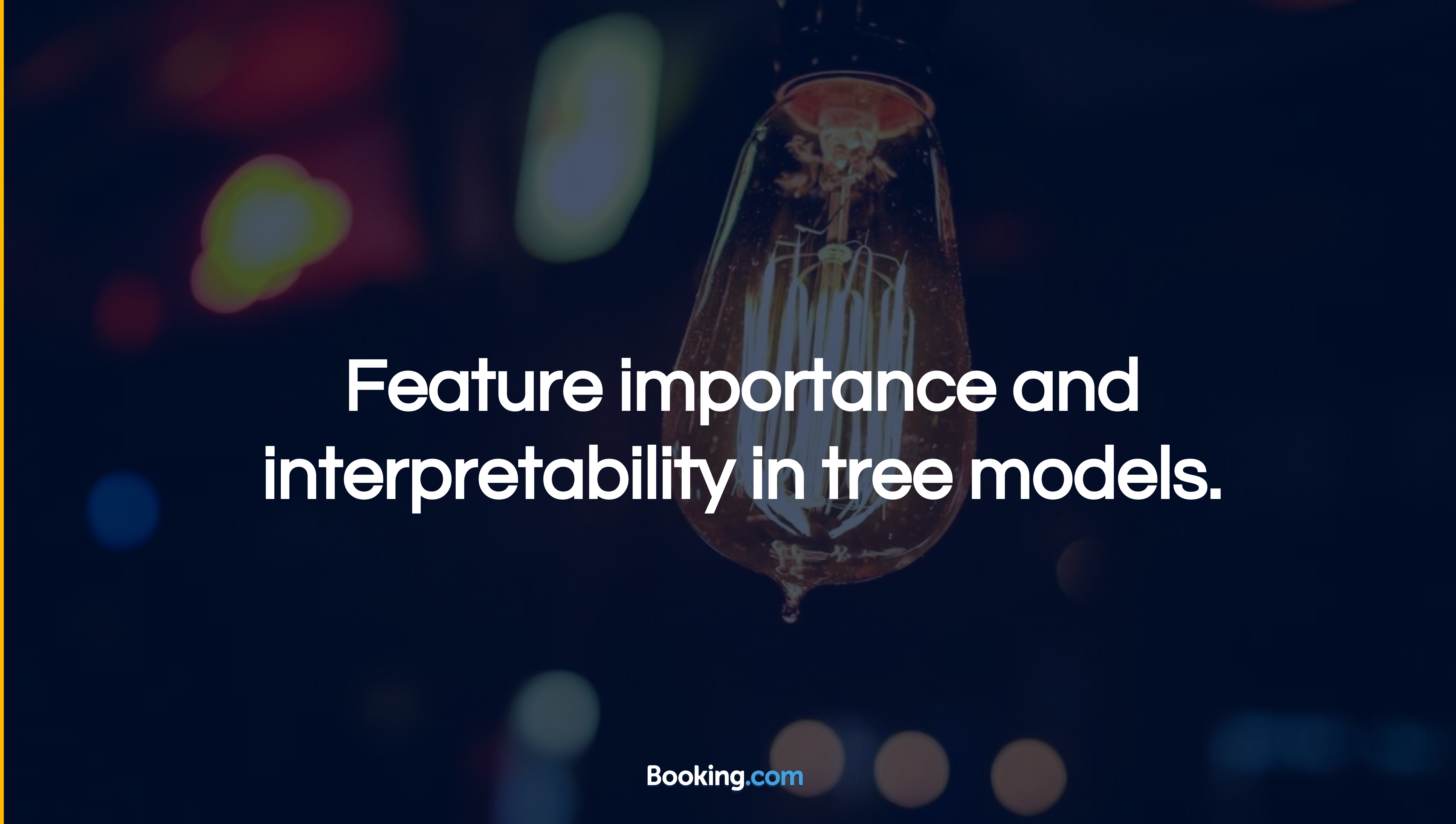
```
train = train.withColumn("fold", F.round(F.rand()*5))
targetEncoder = targetEstimator.fit(train)


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)
```


Feature importance and interpretability in tree models.
Booking.com

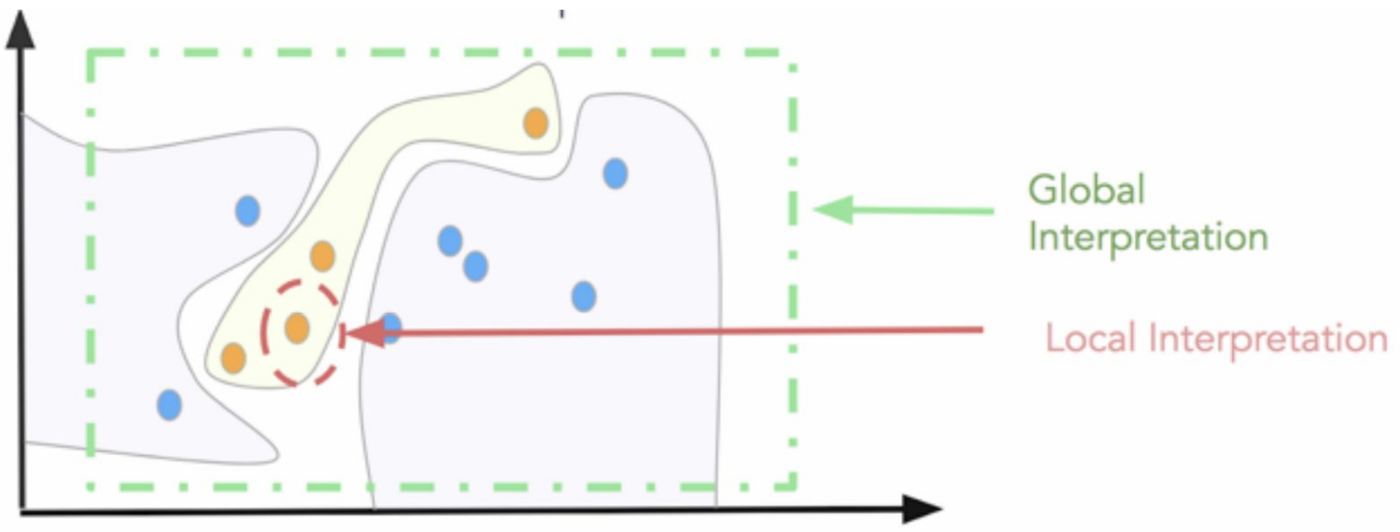
Interpreting a complex model.

Global interpretability

Relationship between the input and the dependent variables.

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 methods.

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* $\phi_i \in \mathbb{R}$*.*

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

3 simple properties uniquely determine additive feature attributions.

1. **Local accuracy**

2. **Missingness**

3. **Consistency**

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

Local accuracy.

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).*

When approximating the original model for a specific input , local accuracy requires the explanation model to at least match the output of the original model for the simplified input

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

Missingness.

Property 2 (Missingness)

$$x'_i = 0 \implies \phi_i = 0 \tag{6}$$

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

If the simplified inputs represent feature presence, then missingness requires features missing in the original input to have no impact.

Missing features have no attributed impact to the model predictions

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

Consistency.

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* $\phi_i(f', x) \geq \phi_i(f, x)$.

Consistency states that if a model changes so that some simplified input's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease.

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

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'*.*

This result implies that methods not based on Shapley values violate local accuracy and/or consistency

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

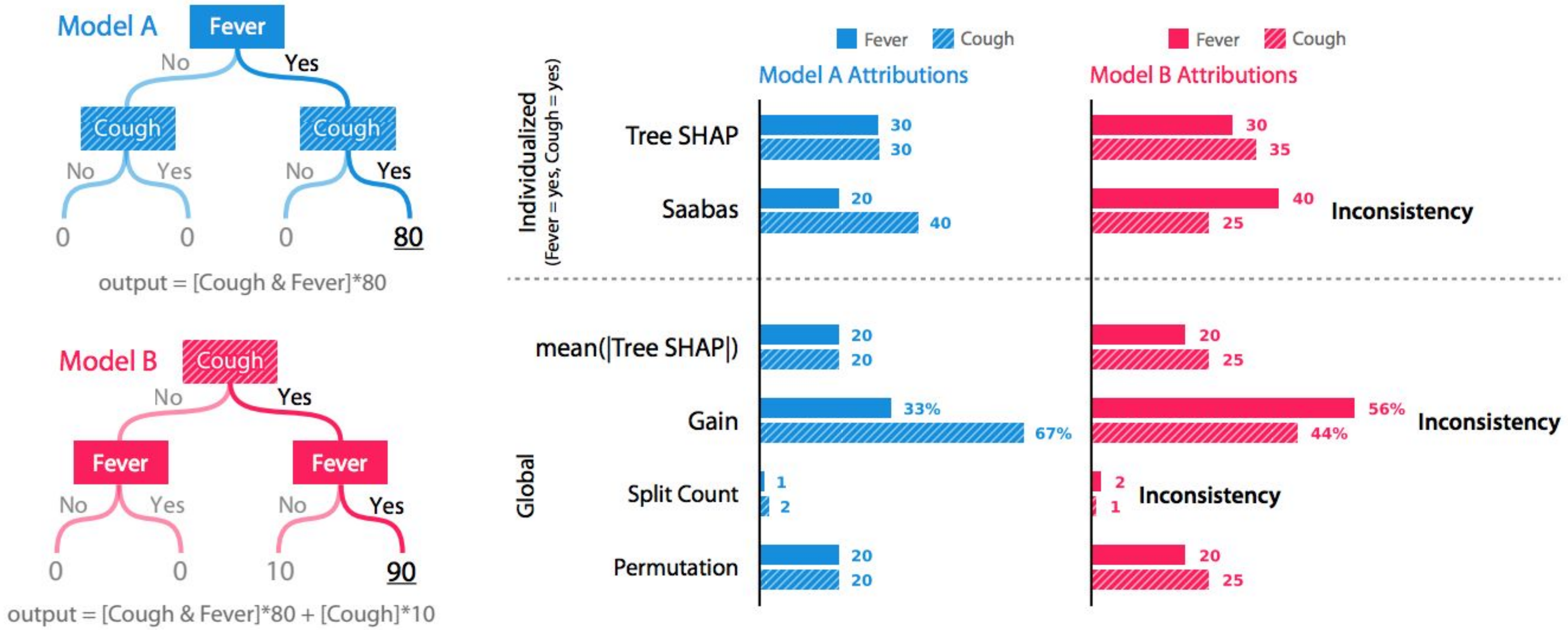
Introducing Shapley values.

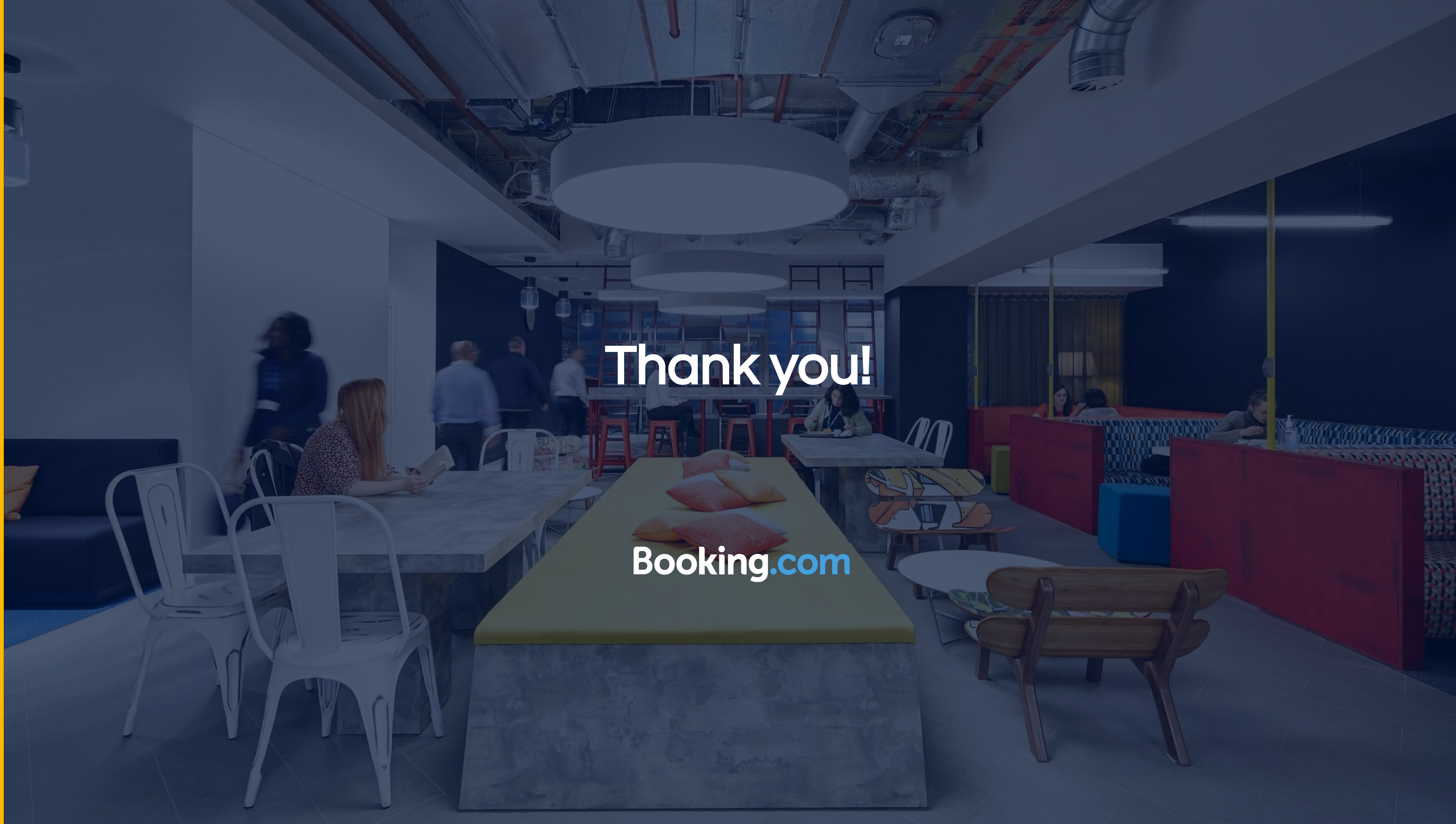
$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right].$$

Importance of i = p(with i) – p(without i)

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

Inconsistencies on Gain and Split count.





Thank you!

Booking.com