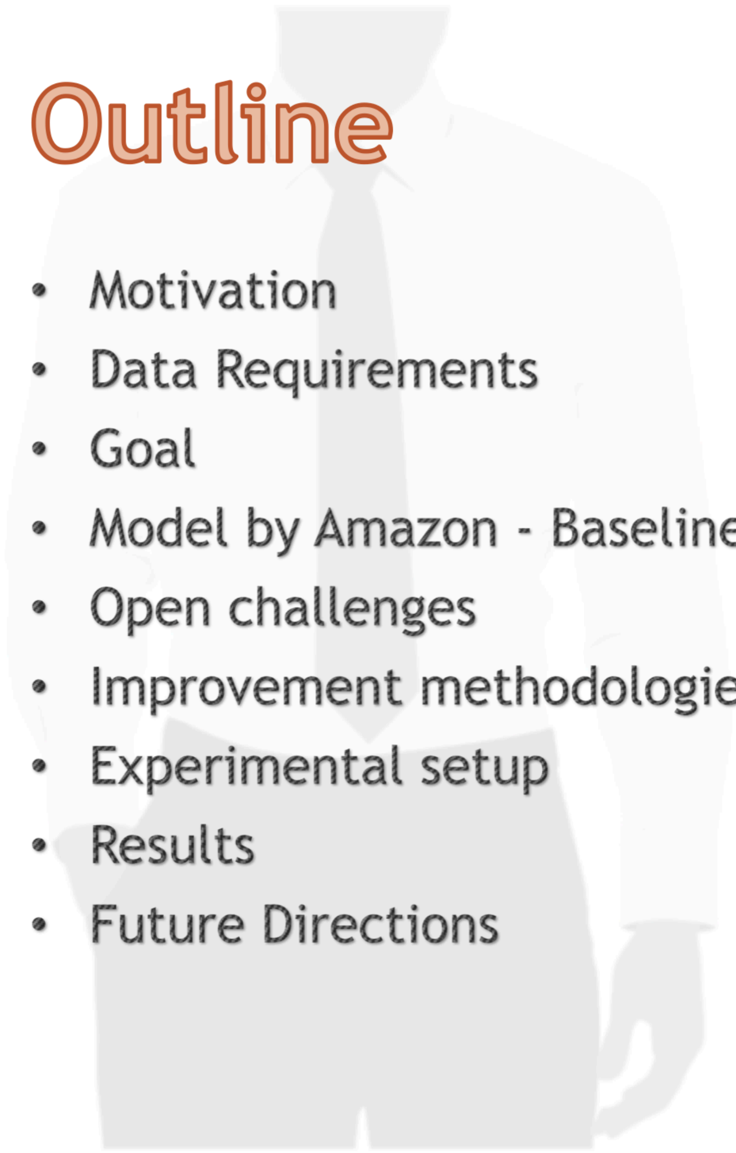
Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces

Using Machine Learning to find fitting sizes of clothing products for customers

TWiML June Americas Meetup

Presentation by: Rishabh Misra

@rishabh_misra_



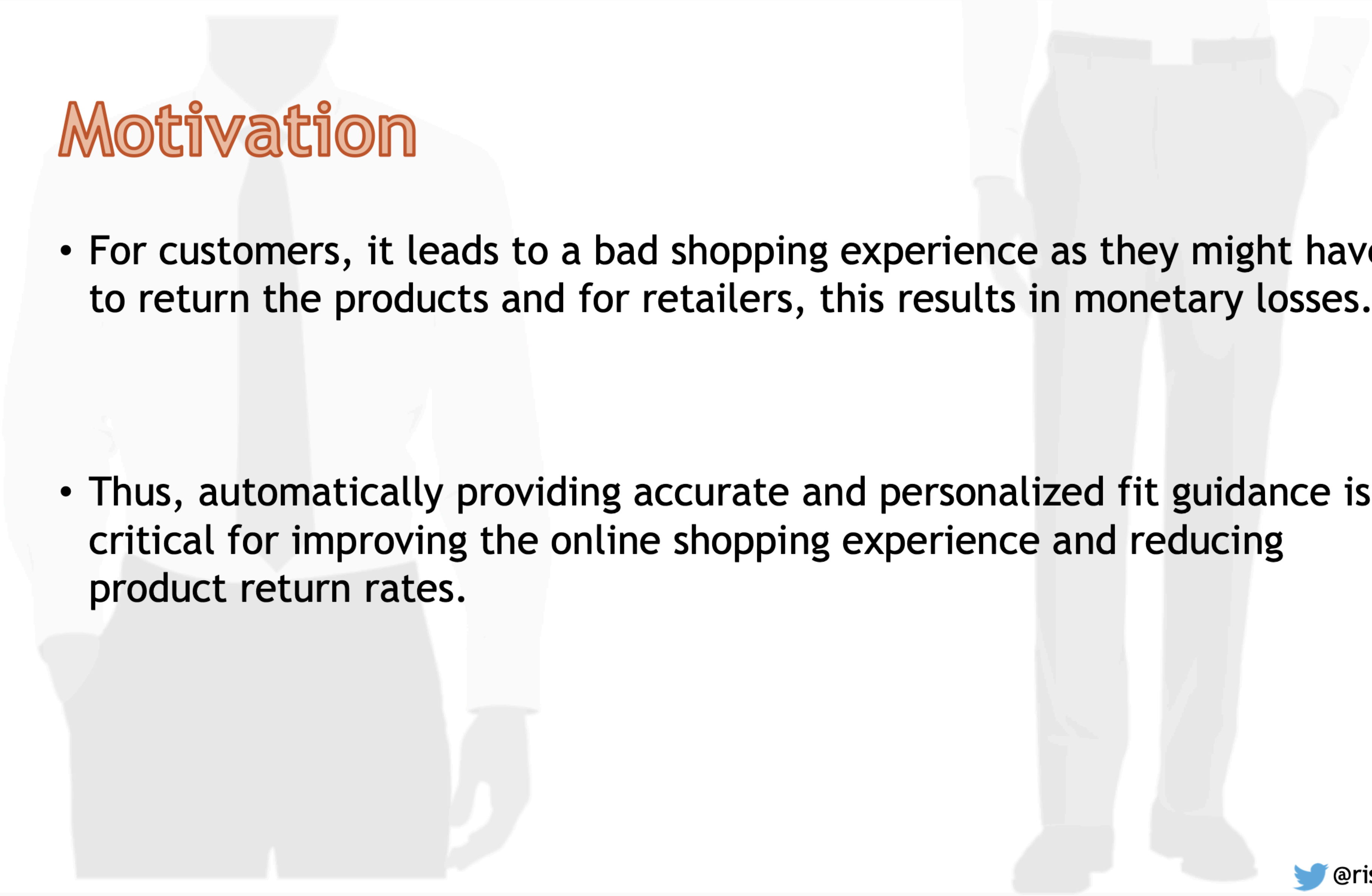
Outline

- Motivation
- Data Requirements
- Goal
- Model by Amazon - Baseline
- Open challenges
- Improvement methodologies
- Experimental setup
- Results
- Future Directions

Motivation

- Who doesn’t love online shopping? It is convenient, offers a wide selection of products and we get great deals.
- Retailers also benefit from the reach that internet provides and get a chance to establish their position in the market.
- However, shopping for clothes online is still tricky owing to *wide sizing variations across different brands* that make it tough for customers to identify proper fitting clothes.

@rishabh_misra_

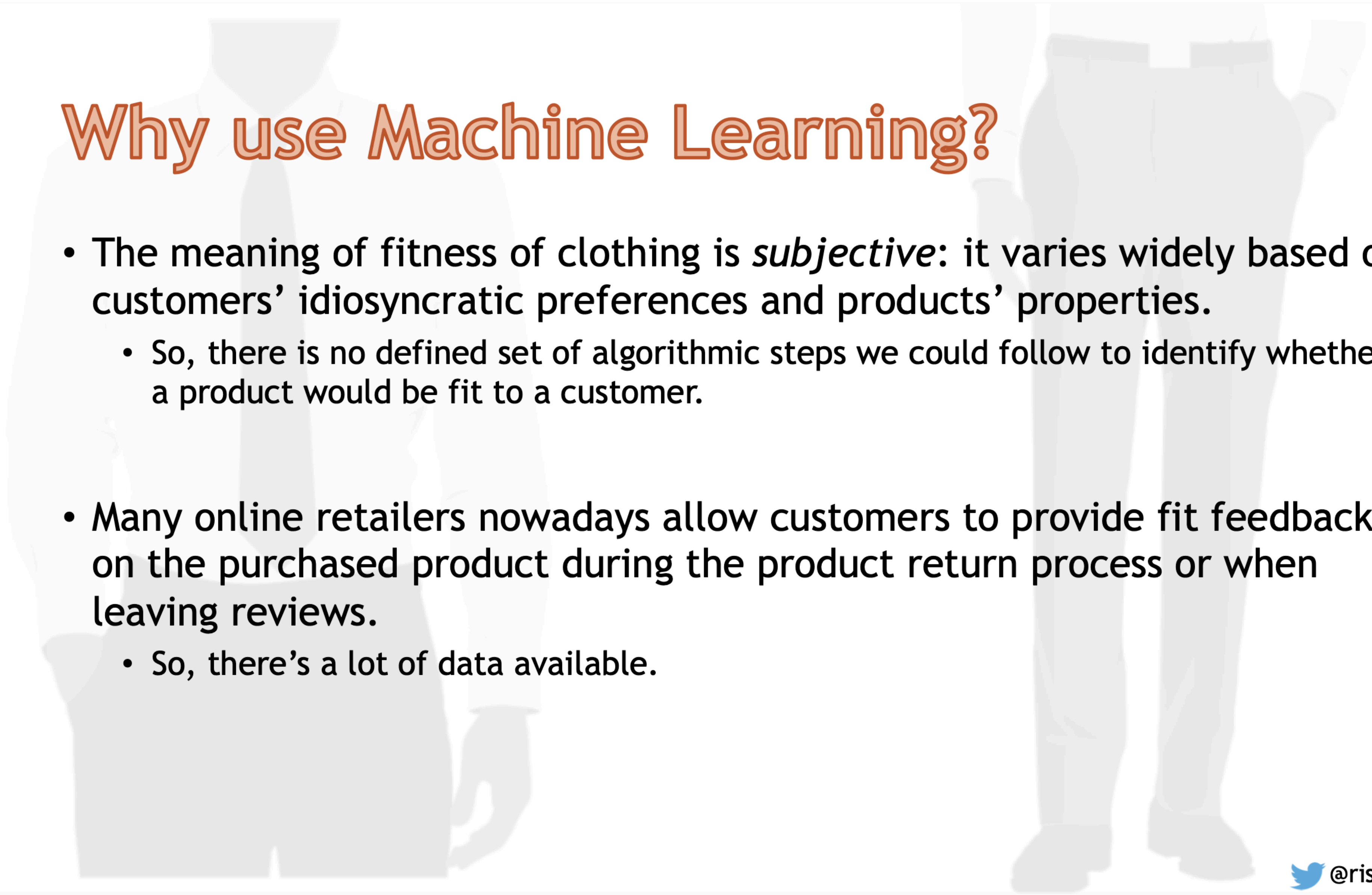


Motivation

- For customers, it leads to a bad shopping experience as they might have to return the products and for retailers, this results in monetary losses.

- Thus, automatically providing accurate and personalized fit guidance is critical for improving the online shopping experience and reducing product return rates.

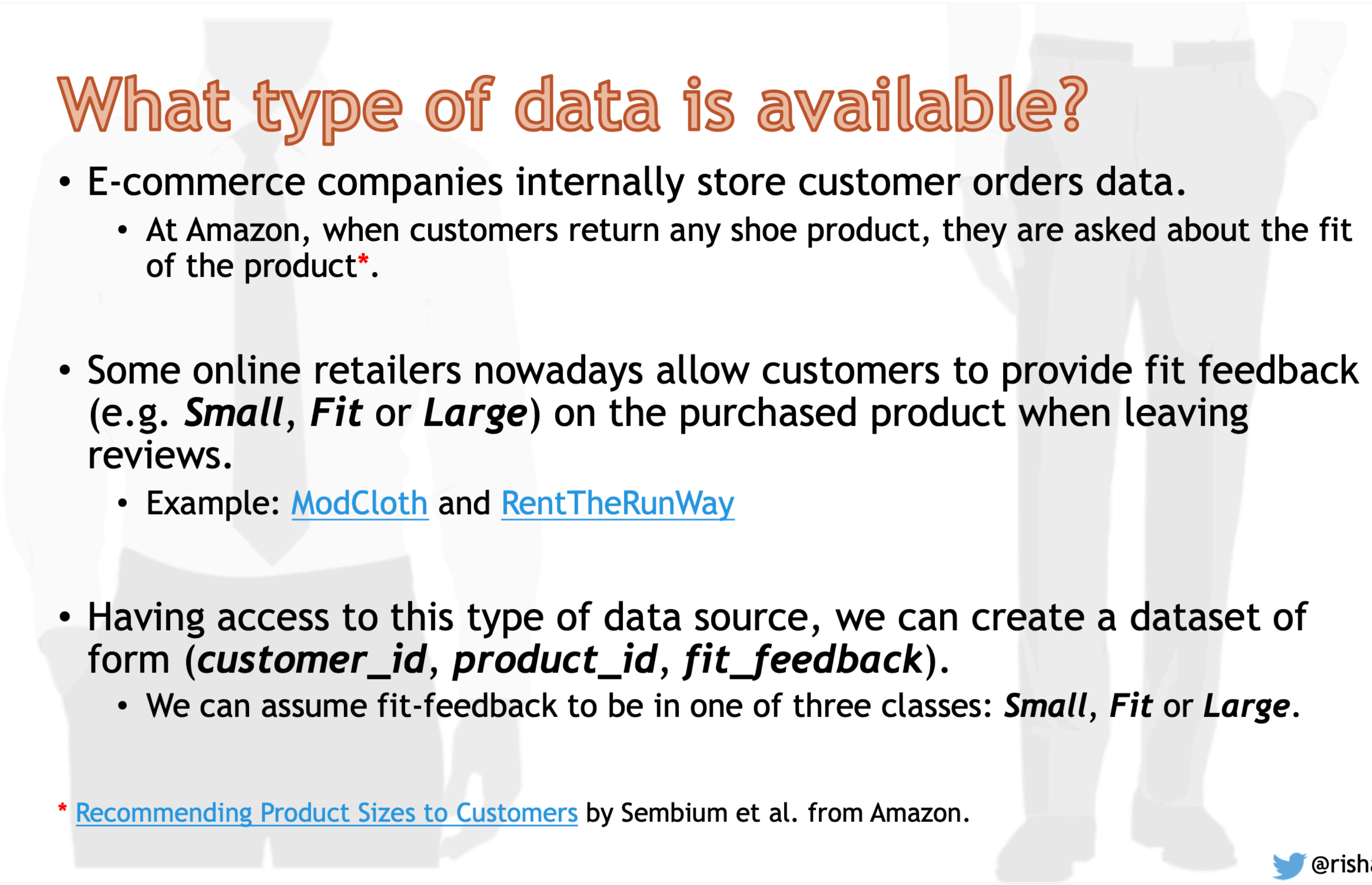
@rishabh_misra_



Why use Machine Learning?

- The meaning of fitness of clothing is *subjective*: it varies widely based on customers' idiosyncratic preferences and products' properties.
  - So, there is no defined set of algorithmic steps we could follow to identify whether a product would be fit to a customer.
- Many online retailers nowadays allow customers to provide fit feedback on the purchased product during the product return process or when leaving reviews.
  - So, there's a lot of data available.

@rishabh_misra_



What type of data is available?

- E-commerce companies internally store customer orders data.
  - At Amazon, when customers return any shoe product, they are asked about the fit of the product*.

- Some online retailers nowadays allow customers to provide fit feedback (e.g. ***Small***, ***Fit*** or ***Large***) on the purchased product when leaving reviews.
  - Example: ModCloth and RentTheRunWay

- Having access to this type of data source, we can create a dataset of form (***customer_id***, ***product_id***, ***fit_feedback***).
  - We can assume fit-feedback to be in one of three classes: ***Small***, ***Fit*** or ***Large***.

* Recommending Product Sizes to Customers by Sembium et al. from Amazon.

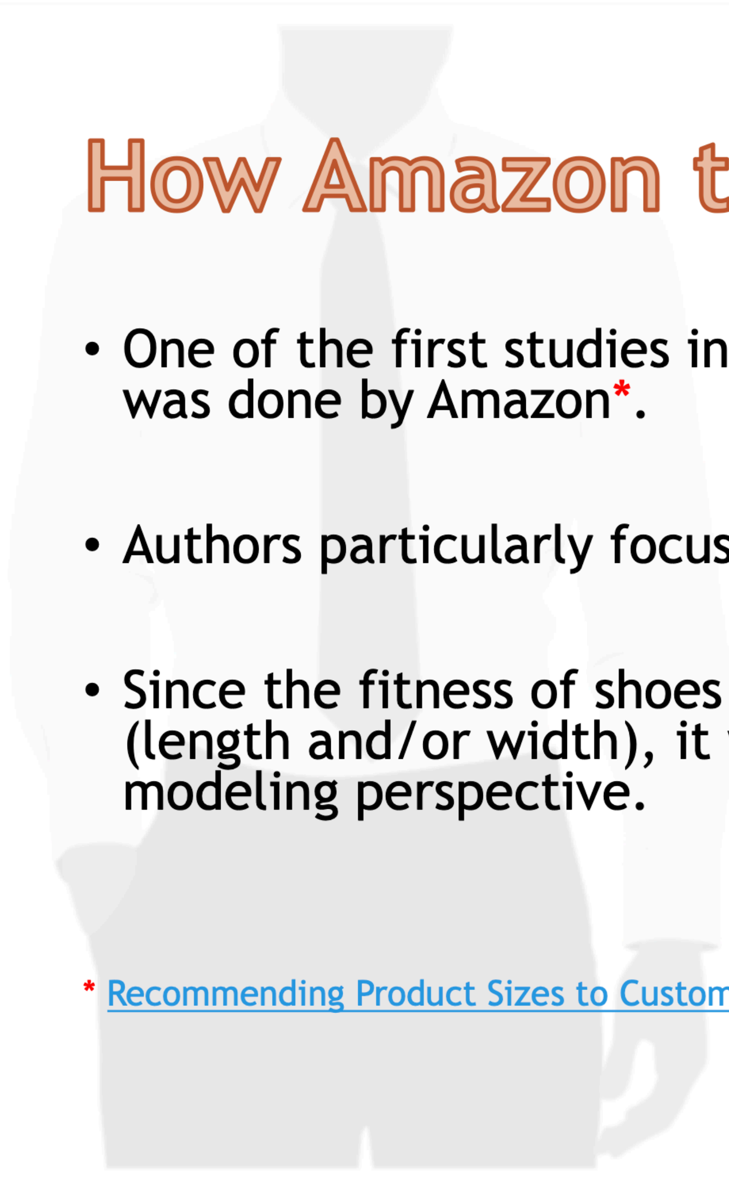
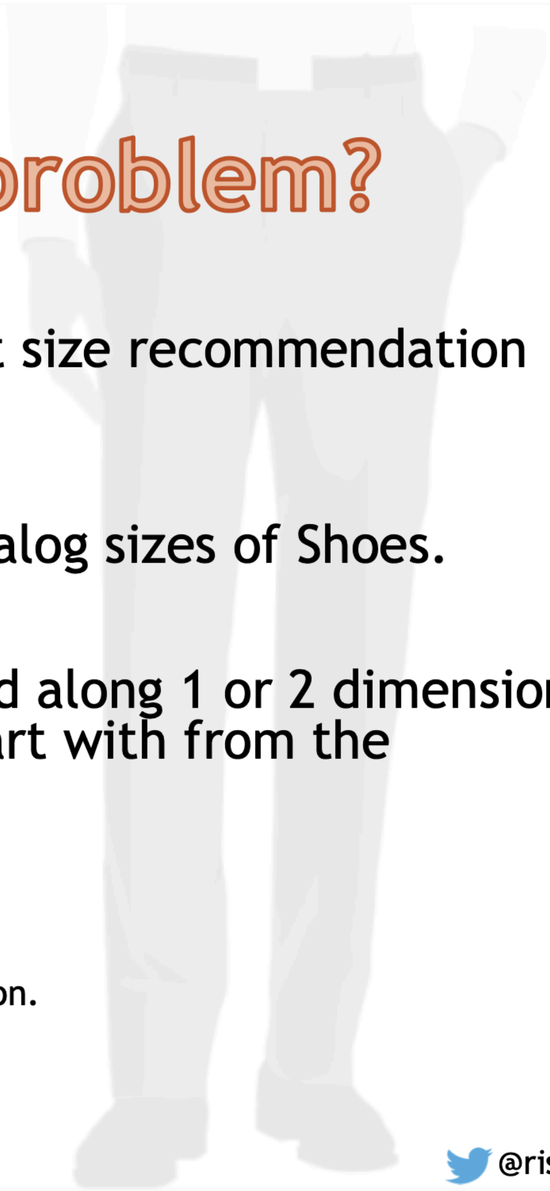
@rishabh_misra_

What's our goal?

- Given the dataset of form (***customer_id***, ***product_id***, ***fit_feedback***), we can define some features for each customer and product.

- Features for customers could capture their preferences of what type of fitting they like. Features for products could capture sizing and style related properties.

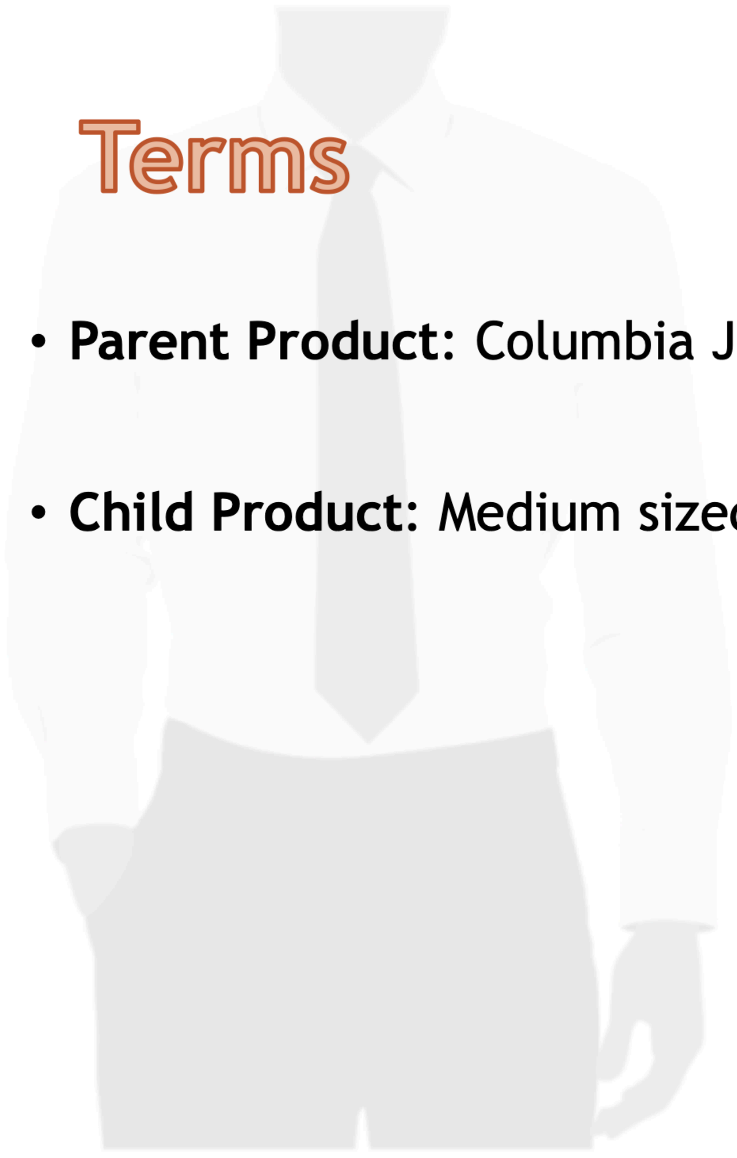
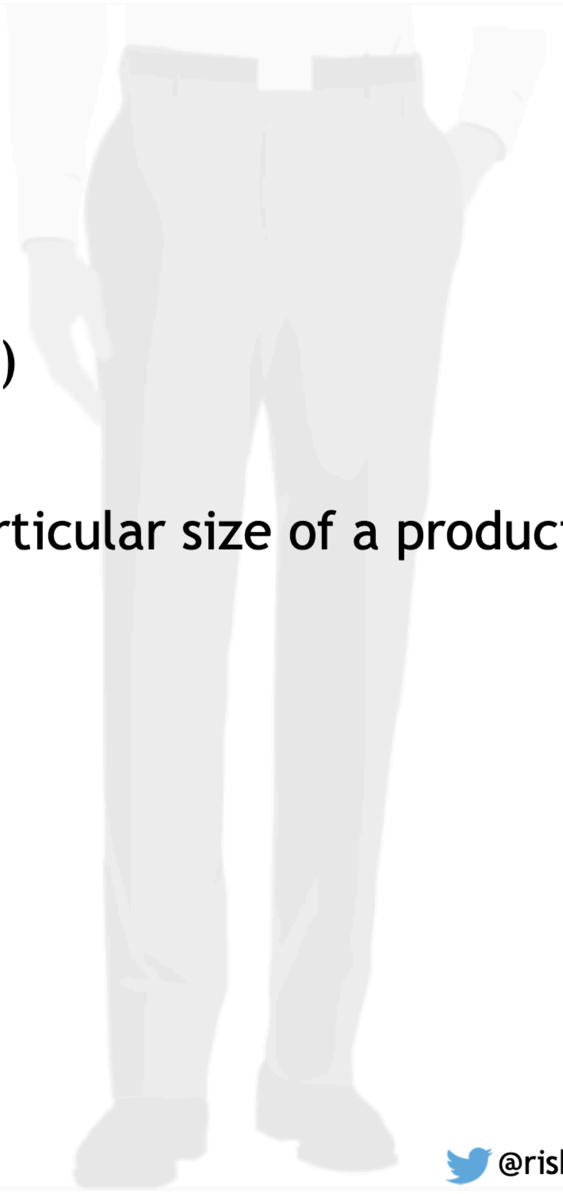
- Our goal can then be defined as:
  - Learn customers' fit preferences and products' size/style related properties from the data.
  - If a customer shows interest in a clothing product by visiting its page, utilize the learned features to recommend them a size that will fit them well.

How Amazon tackled the problem?

- One of the first studies in the domain of product size recommendation was done by Amazon*.
- Authors particularly focus on recommending catalog sizes of Shoes.
- Since the fitness of shoes can arguably be judged along 1 or 2 dimensions (length and/or width), it was a good point to start with from the modeling perspective.

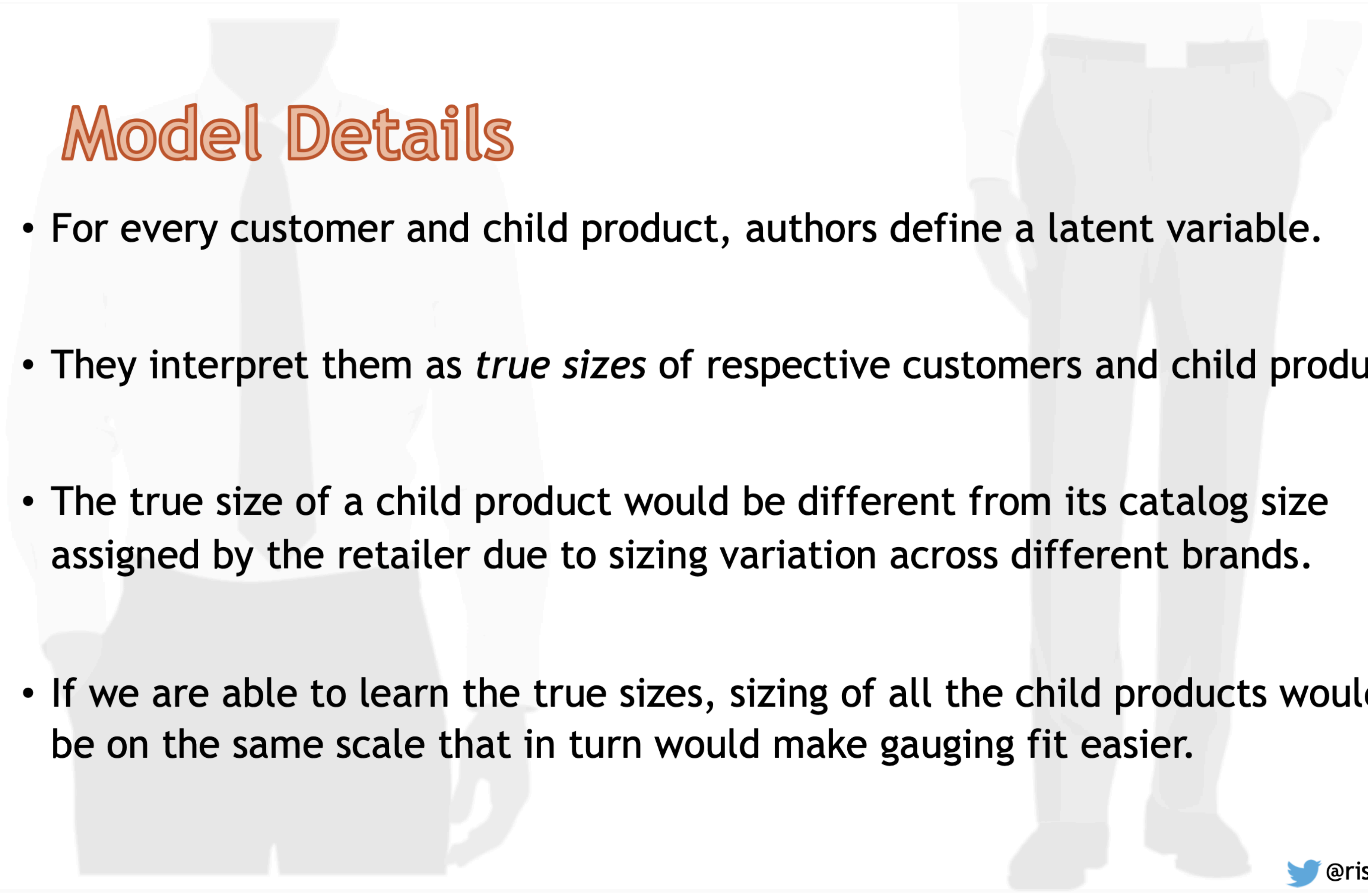
* Recommending Product Sizes to Customers by Sembium et al. from Amazon.

Terms

- **Parent Product:** Columbia Jacket (a product itself)
- **Child Product:** Medium sized Columbia Jacket (particular size of a product)

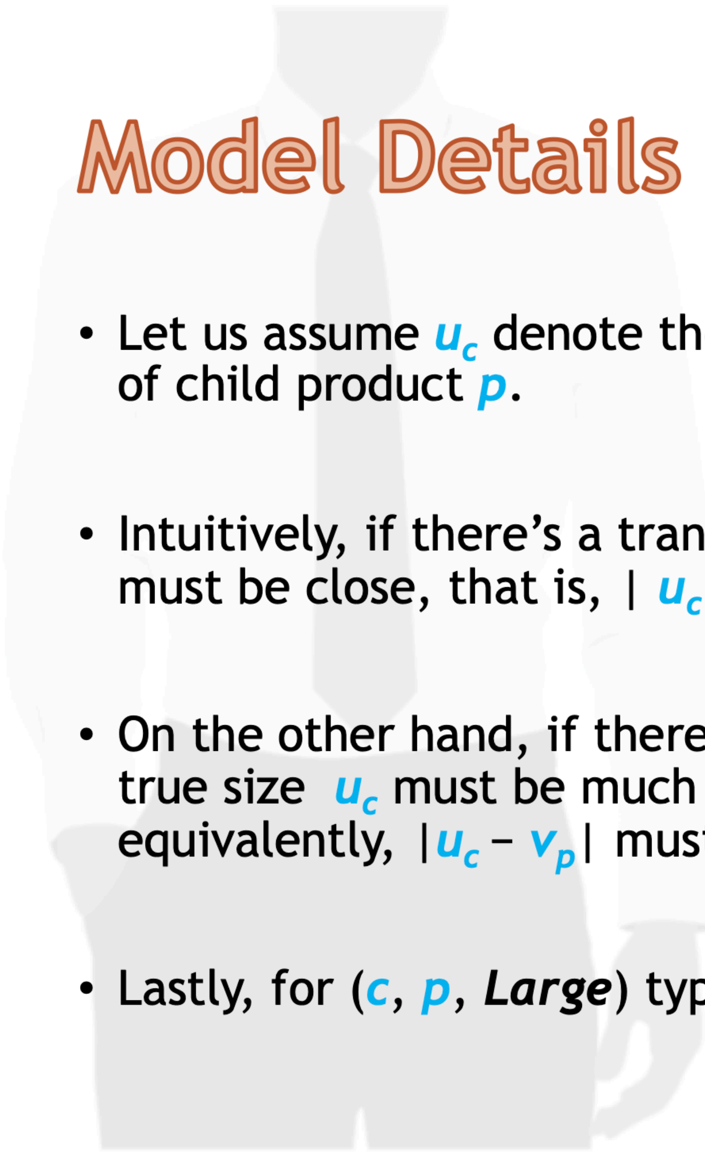
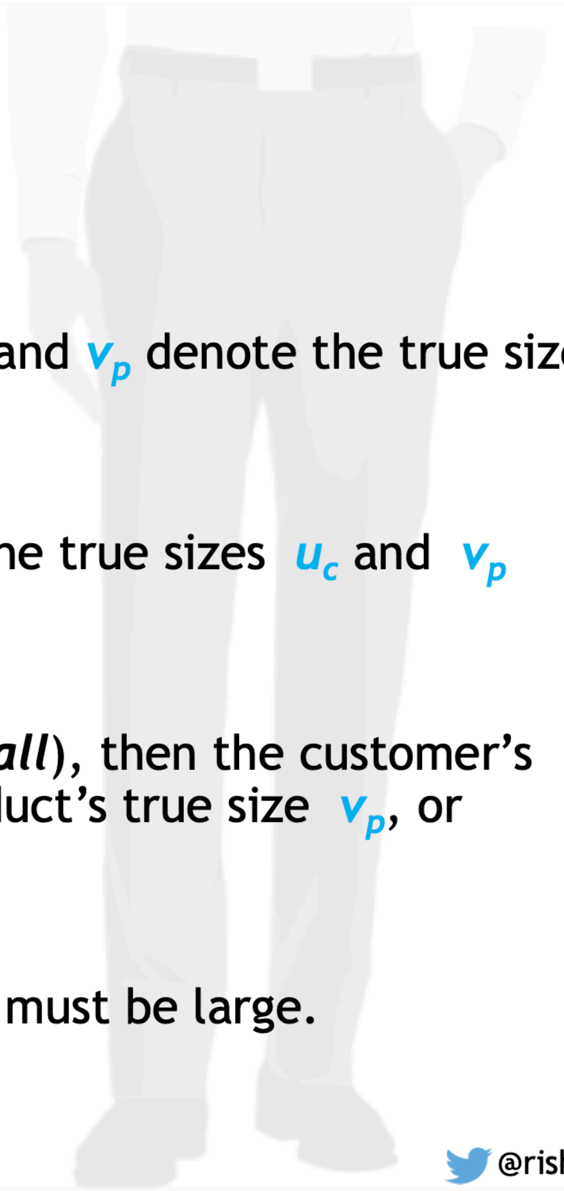
@rishabh_misra_



Model Details

- For every customer and child product, authors define a latent variable.
- They interpret them as *true sizes* of respective customers and child products.
- The true size of a child product would be different from its catalog size assigned by the retailer due to sizing variation across different brands.
- If we are able to learn the true sizes, sizing of all the child products would be on the same scale that in turn would make gauging fit easier.

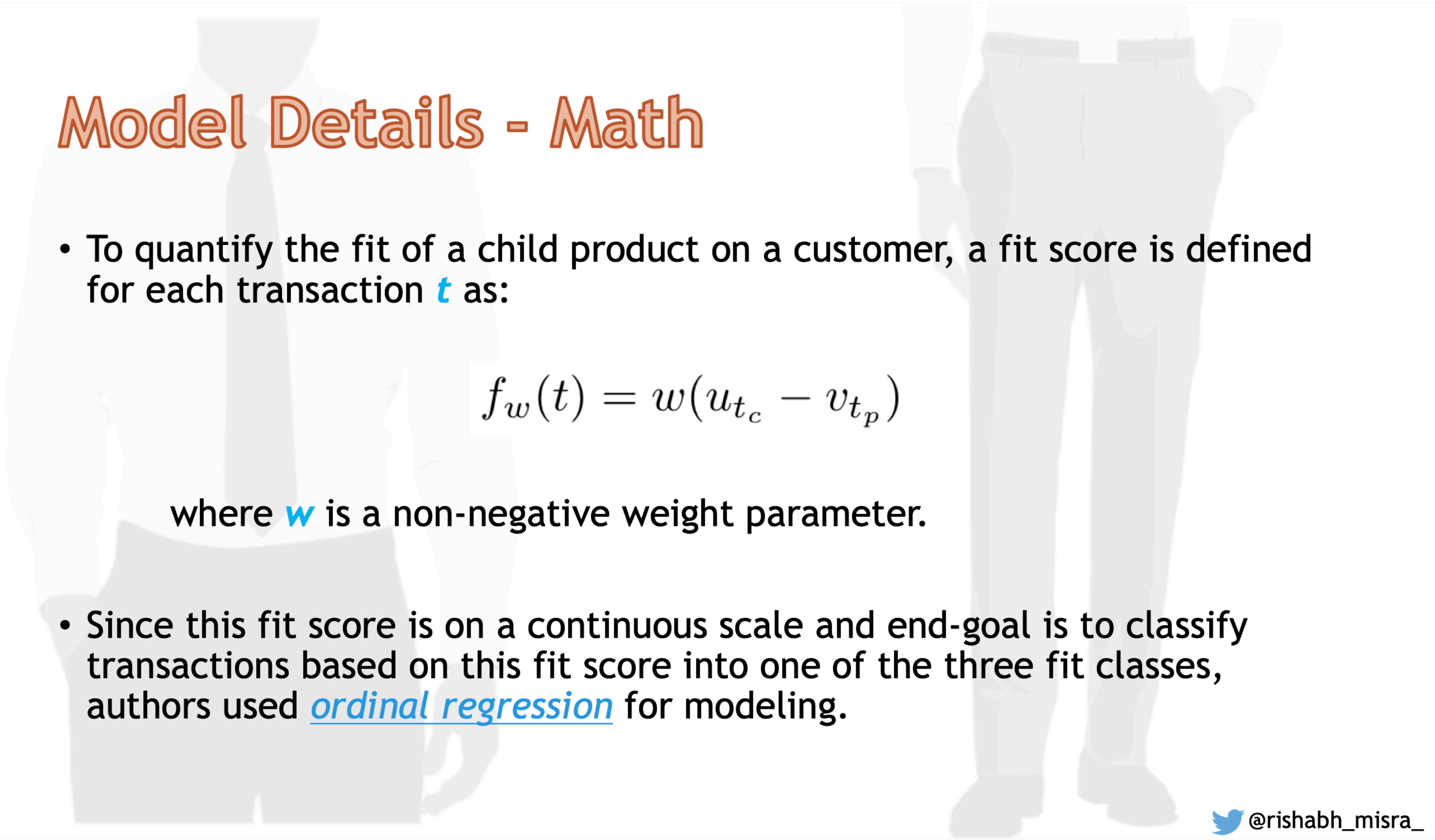
@rishabh_misra_

Model Details - Math

- Let us assume u_c denote the true size of customer c and v_p denote the true size of child product p.
- Intuitively, if there's a transaction (c, p, ***Fit***), then the true sizes u_c and v_p must be close, that is, $| u_c - v_p |$ must be small.
- On the other hand, if there's a transaction (c, p, ***Small***), then the customer's true size u_c must be much larger than the child product's true size v_p, or equivalently, $|u_c - v_p|$ must be large.
- Lastly, for (c, p, ***Large***) type of transactions, $v_p - u_c$ must be large.

@rishabh_misra_



Model Details - Math

- To quantify the fit of a child product on a customer, a fit score is defined for each transaction ***t*** as:

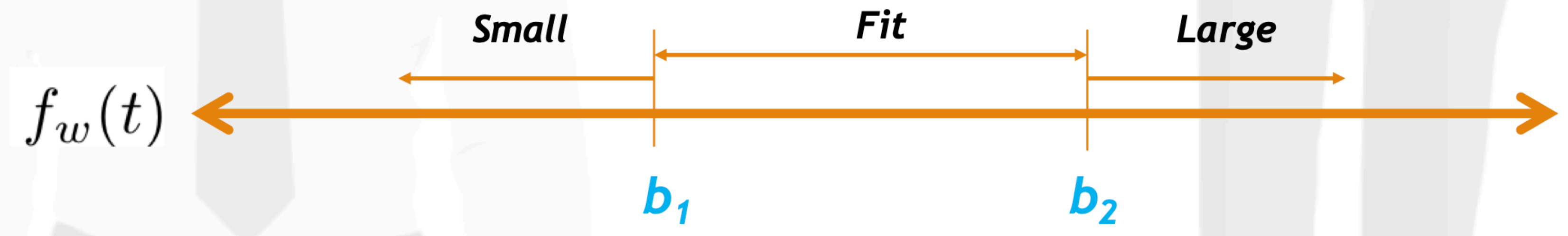
$$f_w(t) = w(u_{t_c} - v_{t_p})$$

where ***w*** is a non-negative weight parameter.

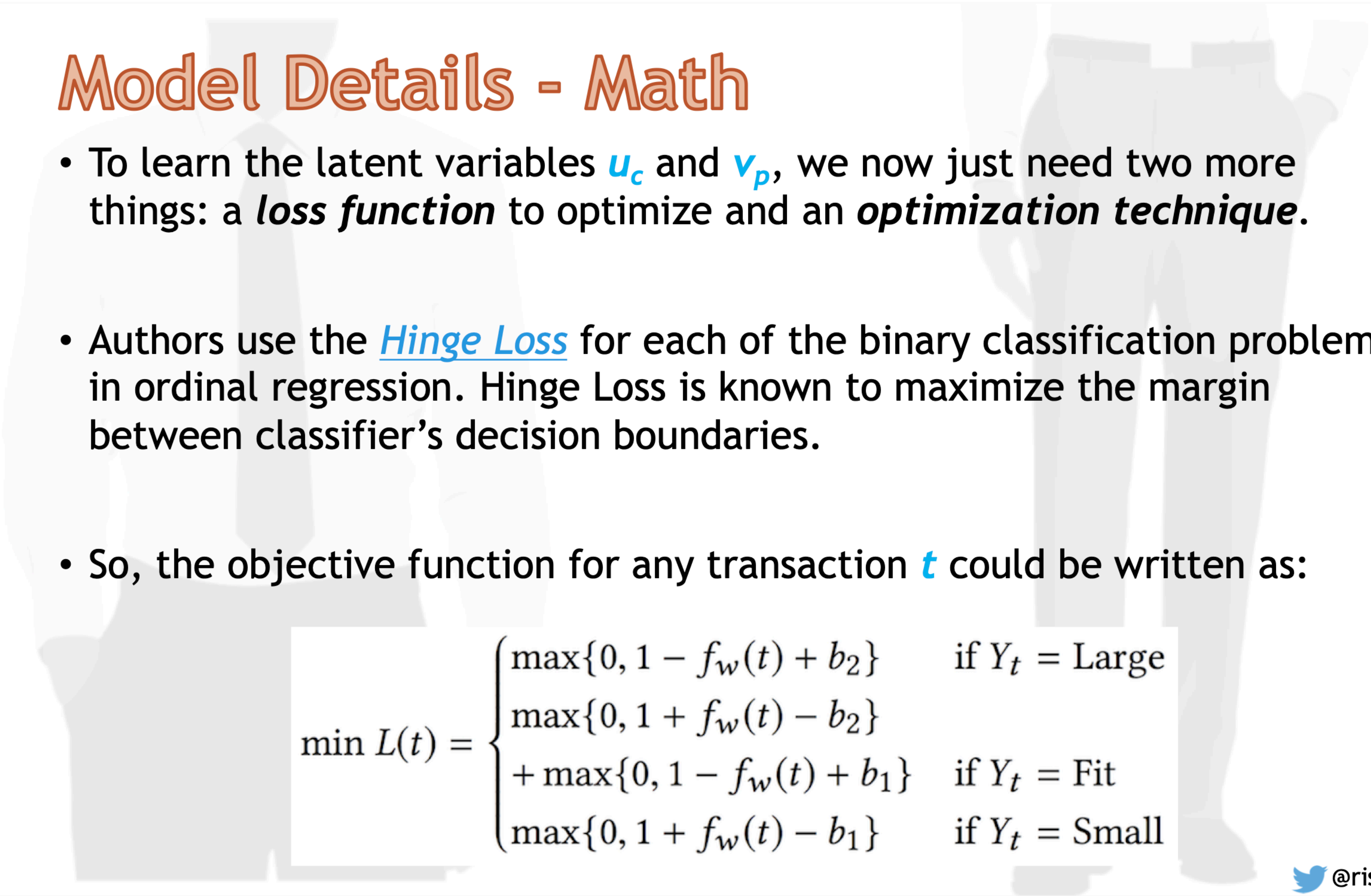
- Since this fit score is on a continuous scale and end-goal is to classify transactions based on this fit score into one of the three fit classes, authors used *ordinal regression* for modeling.

@rishabh_misra_

Model Details - Math



@rishabh_misra_



Model Details - Math

- To learn the latent variables u_c and v_p, we now just need two more things: a ***loss function*** to optimize and an ***optimization technique***.

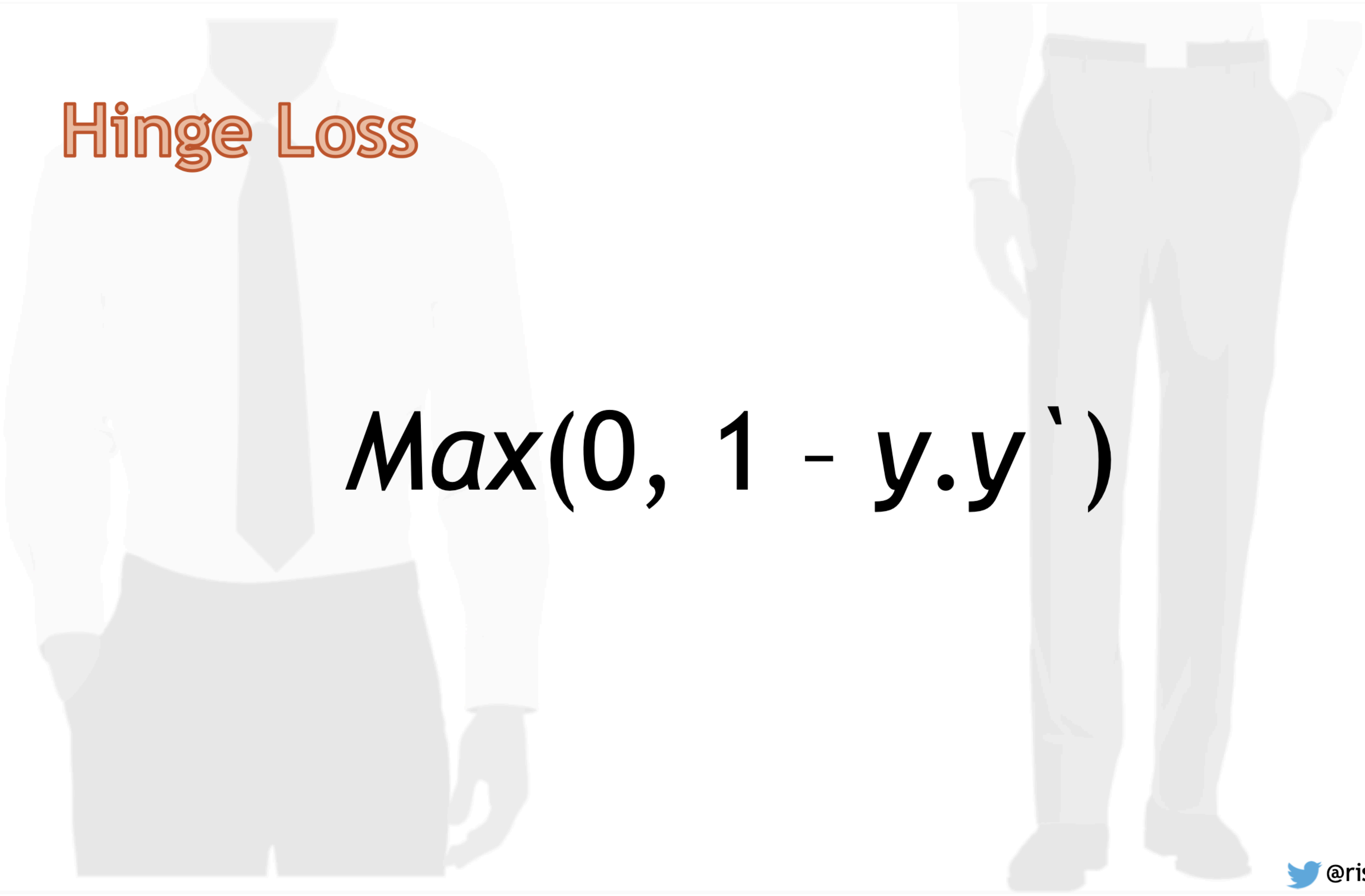
- Authors use the *Hinge Loss* for each of the binary classification problems in ordinal regression. Hinge Loss is known to maximize the margin between classifier's decision boundaries.

- So, the objective function for any transaction t could be written as:

$$\min L(t) = \begin{cases} \max\{0, 1 - f_w(t) + b_2\} & \text{if } Y_t = \text{Large} \\ \max\{0, 1 + f_w(t) - b_2\} & \\ + \max\{0, 1 - f_w(t) + b_1\} & \text{if } Y_t = \text{Fit} \\ \max\{0, 1 + f_w(t) - b_1\} & \text{if } Y_t = \text{Small} \end{cases}$$

@rishabh_misra_

Hinge Loss



$$Max(0, 1 - y.y`)$$

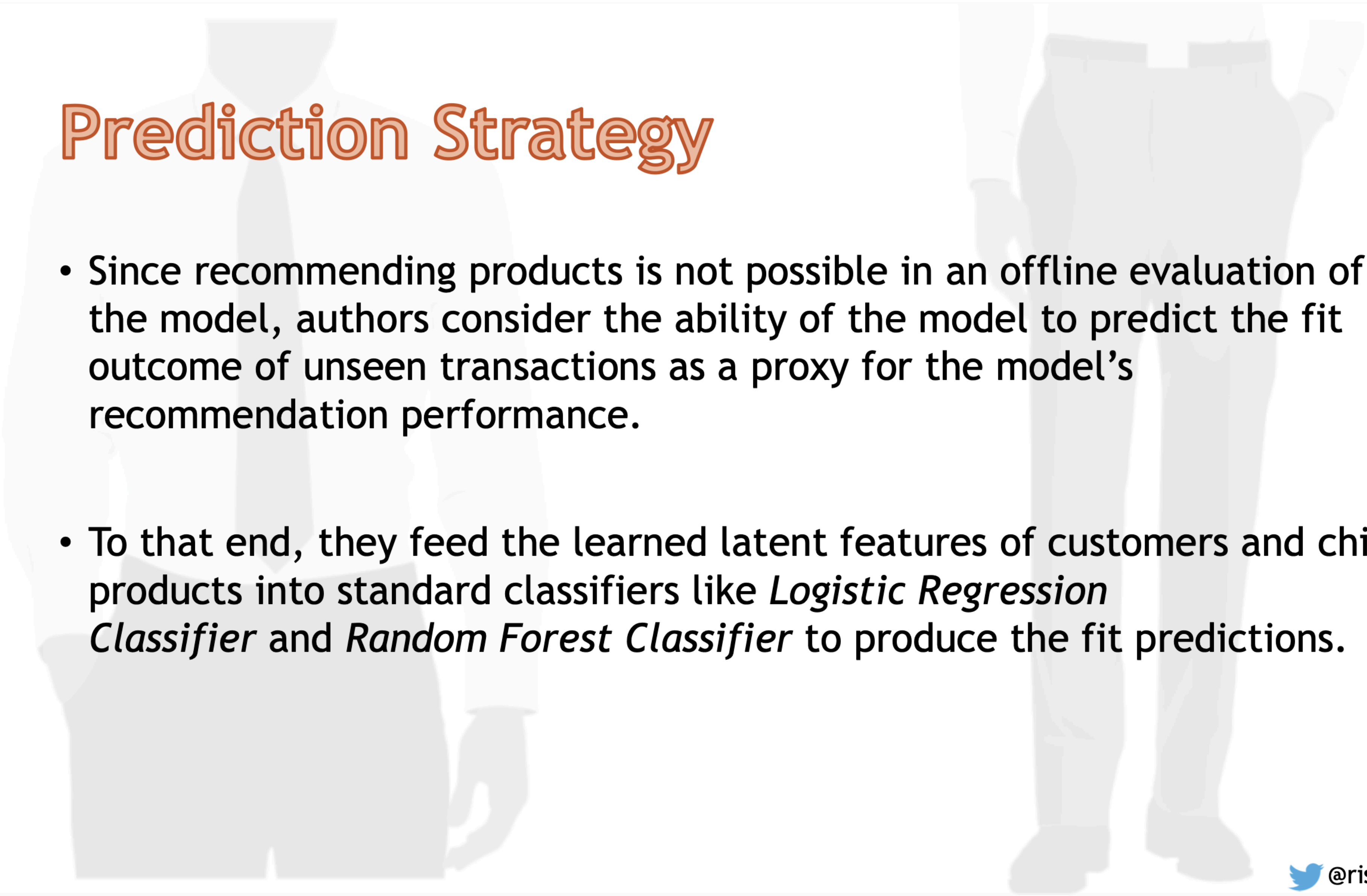
@rishabh_misra_

Model Details - Math

- The overall loss is simply the sum of losses for each transaction.

- It is minimized when for any transaction t with a fit outcome Y_t:
  - $f_w(t) > b_2$ if Y_t = ***Large***,
  - $f_w(t) < b_1$ if Y_t = ***Small***, and
  - $b_1 < f_w(t) < b_2$ if Y_t = ***Fit***.

- Authors use *Stochastic Gradient Descent* to optimize the objective.

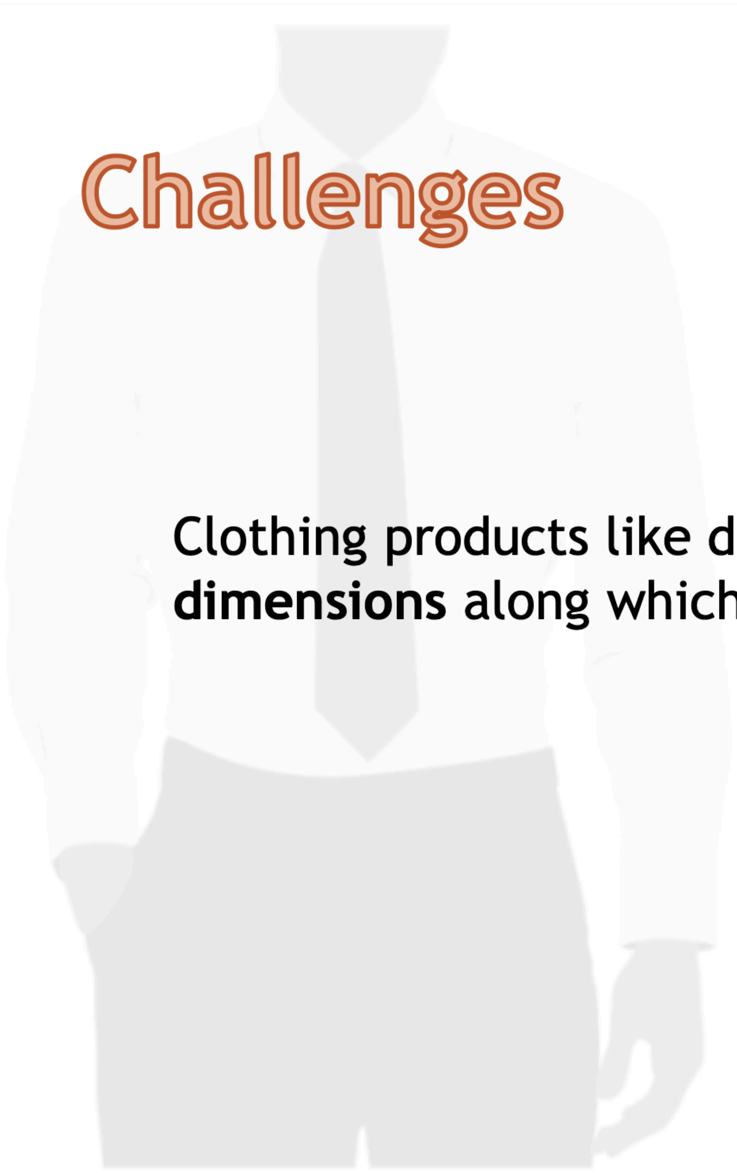


Prediction Strategy

- Since recommending products is not possible in an offline evaluation of the model, authors consider the ability of the model to predict the fit outcome of unseen transactions as a proxy for the model's recommendation performance.

- To that end, they feed the learned latent features of customers and child products into standard classifiers like *Logistic Regression Classifier* and *Random Forest Classifier* to produce the fit predictions.

Challenges



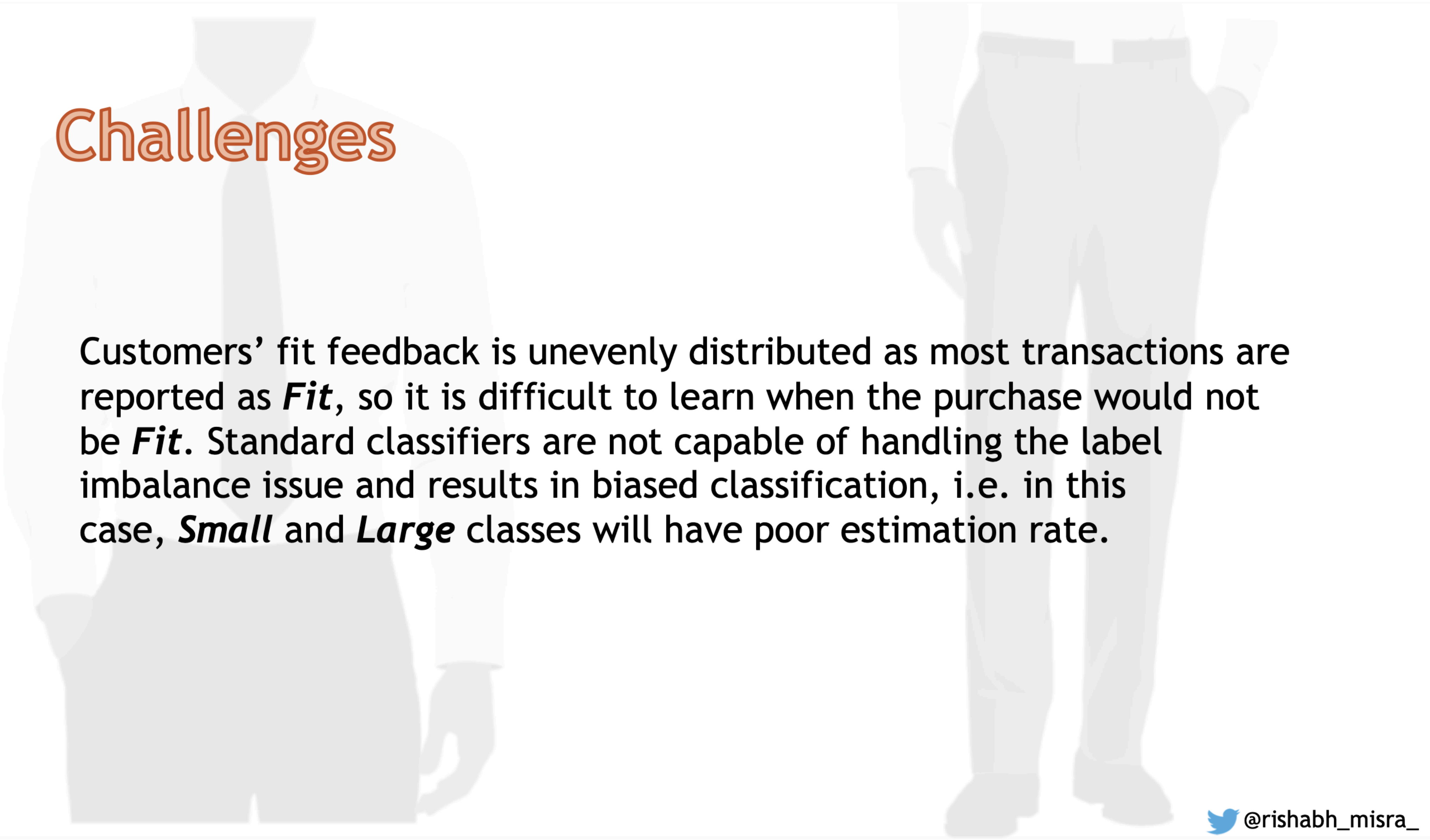
Clothing products like dresses and shirts have relatively **more dimensions** along which the fit is determined.

Challenges

Customers' fit preference might **vary across different product categories** for each customer, for example, customers might prefer a jacket to be a little loose whereas a wet suit to be more form fitting. Thus, a single latent feature for each child product and customer might not be enough to capture all the variability in the data.

Challenges

Customers' fit feedback is unevenly distributed as most transactions are reported as ***Fit***, so it is difficult to learn when the purchase would not be ***Fit***. Standard classifiers are not capable of handling the label imbalance issue and results in biased classification, i.e. in this case, ***Small*** and ***Large*** classes will have poor estimation rate.



Challenges

No publicly available dataset.

@rishabh_misra_

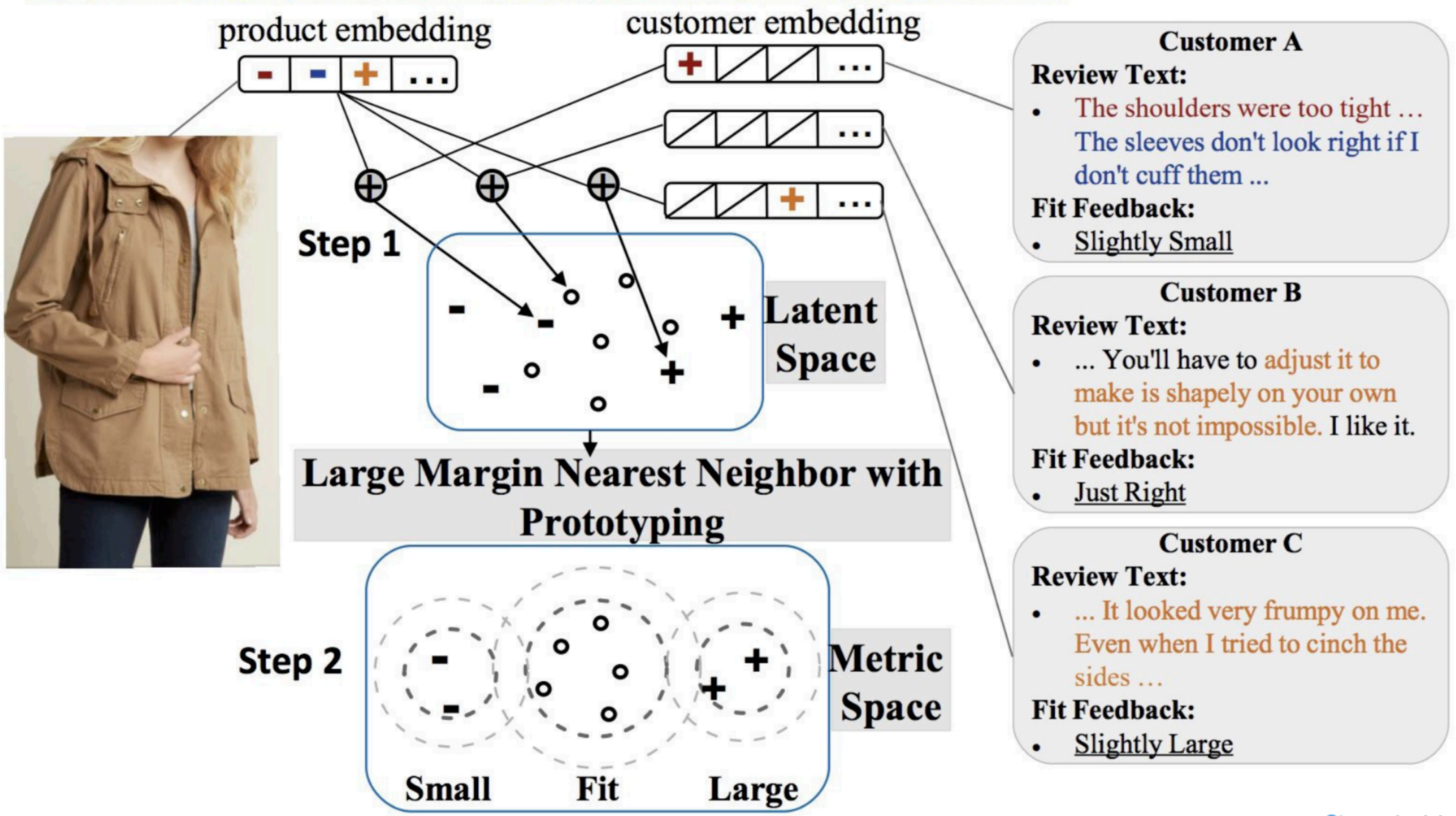
How can we improve?

- To address the first two challenges, we consider **multiple latent features** for each customer and child product. Intuitively, this enables us to capture rich fit preferences of customers on various product aspects (like shoulders, waist etc.).
- To address the third challenge, we take the help of a **metric learning technique with prototyping.**
- We **collected** two novel **datasets** from the web to address the fourth challenge.

@rishabh_misra_

Jacket in Maplewood (Size: Medium)

https://www.modcloth.com/shop/outerwear/woods-you-be-mine-jacket-in-maplewood/118317.html



Customer A

Review Text:

- The shoulders were too tight … The sleeves don't look right if I don't cuff them ...

Fit Feedback:

- Slightly Small

Customer B

Review Text:

- ... You'll have to adjust it to make is shapely on your own but it's not impossible. I like it.

Fit Feedback:

- Just Right

Customer C

Review Text:

- ... It looked very frumpy on me. Even when I tried to cinch the sides …

Fit Feedback:

- Slightly Large

@rishabh_misra_

Learning Semantics of Fit

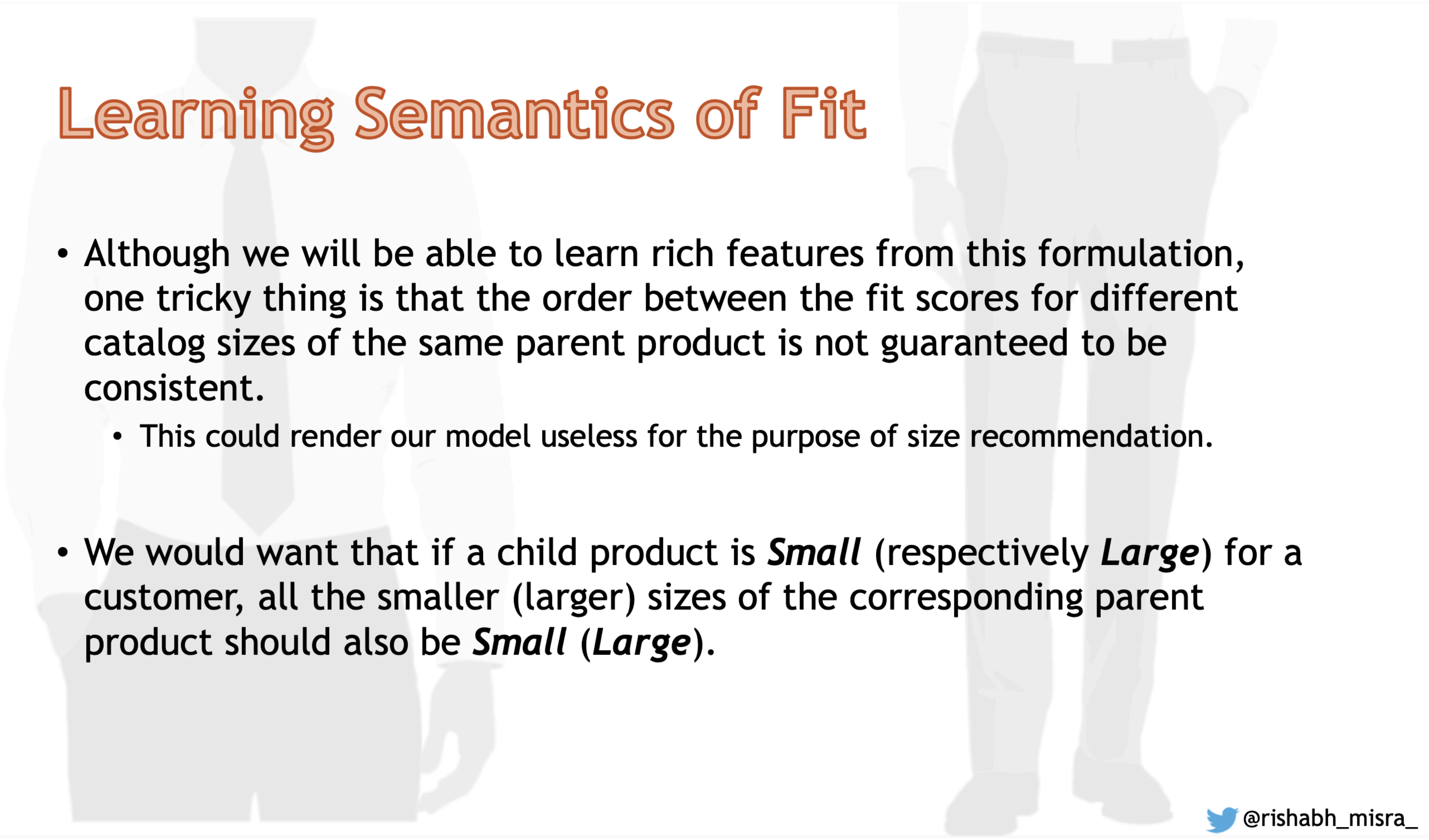
- We decompose the fit feedback signal using a ***latent factor model*** formulation (which ultimately helps us extract more informative features from the data).

$$f_w(t) = \Big\langle \underbrace{\mathbf{w}}_{\text{weight vector}}, \overbrace{\alpha \oplus b_{t_c} \oplus b_{t_{pp}}}^{\text{fit bias terms}} \oplus \underbrace{(\mathbf{u}_{t_c} \odot \mathbf{v}_{t_p})}_{\text{fit compatibility}} \Big\rangle$$

where ***pp*** denotes parent product, ***u*** and ***v*** are **k**-dimensional latent features, ***α*** is a global bias term, ⊕ denotes concatenation and ⊙ denotes element-wise product.

- The bias term b_{t_pp} captures the notion that certain parent products tend to be reported more unfit because of their inherent features/build whereas the bias term b_{t_c} captures the notion that certain customers are highly sensitive to fit while others could be more accommodating.

@rishabh_misra_



Learning Semantics of Fit

- Although we will be able to learn rich features from this formulation, one tricky thing is that the order between the fit scores for different catalog sizes of the same parent product is not guaranteed to be consistent.
  - This could render our model useless for the purpose of size recommendation.

- We would want that if a child product is ***Small*** (respectively ***Large***) for a customer, all the smaller (larger) sizes of the corresponding parent product should also be ***Small*** (***Large***).

@rishabh_misra_

$f_w(t)$ $<$ $f_w(t)$ $<$ $f_w(t)$

@rishabh_misra_

Learning Semantics of Fit

- Observing fit score equation, we see that fit score for a parent product *pp* and a customer *c* only varies based on child product's latent features v_p.

$$f_w(t) = \left\langle \underbrace{\mathbf{w}}_{\text{weight vector}}, \overbrace{\alpha \oplus b_{t_c} \oplus b_{t_{pp}}}^{\text{fit bias terms}} \oplus \underbrace{(\mathbf{u}_{\mathbf{t_c}} \odot \mathbf{v}_{\mathbf{t_p}})}_{\text{fit compatibility}} \right\rangle$$

- So to resolve this issue, it would be enough to enforce that all the latent features of a child product *p* are strictly larger (smaller) than the latent features of next smaller (larger) catalog product p^- (p^+) if a smaller (larger) size exists.

@rishabh_misra_

Learning Semantics of Fit

- So the modified objective function could be stated as follows:

$$\min L(t) = \begin{cases} \max\{0, 1 - f_w(t) + b_2\} & \text{if } Y_t = \text{Large} \\ \max\{0, 1 + f_w(t) - b_2\} & \\ + \max\{0, 1 - f_w(t) + b_1\} & \text{if } Y_t = \text{Fit} \\ \max\{0, 1 + f_w(t) - b_1\} & \text{if } Y_t = \text{Small} \end{cases}$$

$$\textbf{s.t. } \mathbf{v}_{\mathbf{t}_{\mathbf{p}^-}} < \mathbf{v}_{\mathbf{t}_{\mathbf{p}}} < \mathbf{v}_{\mathbf{t}_{\mathbf{p}^+}} .$$

@rishabh_misra_

Learning Semantics of Fit

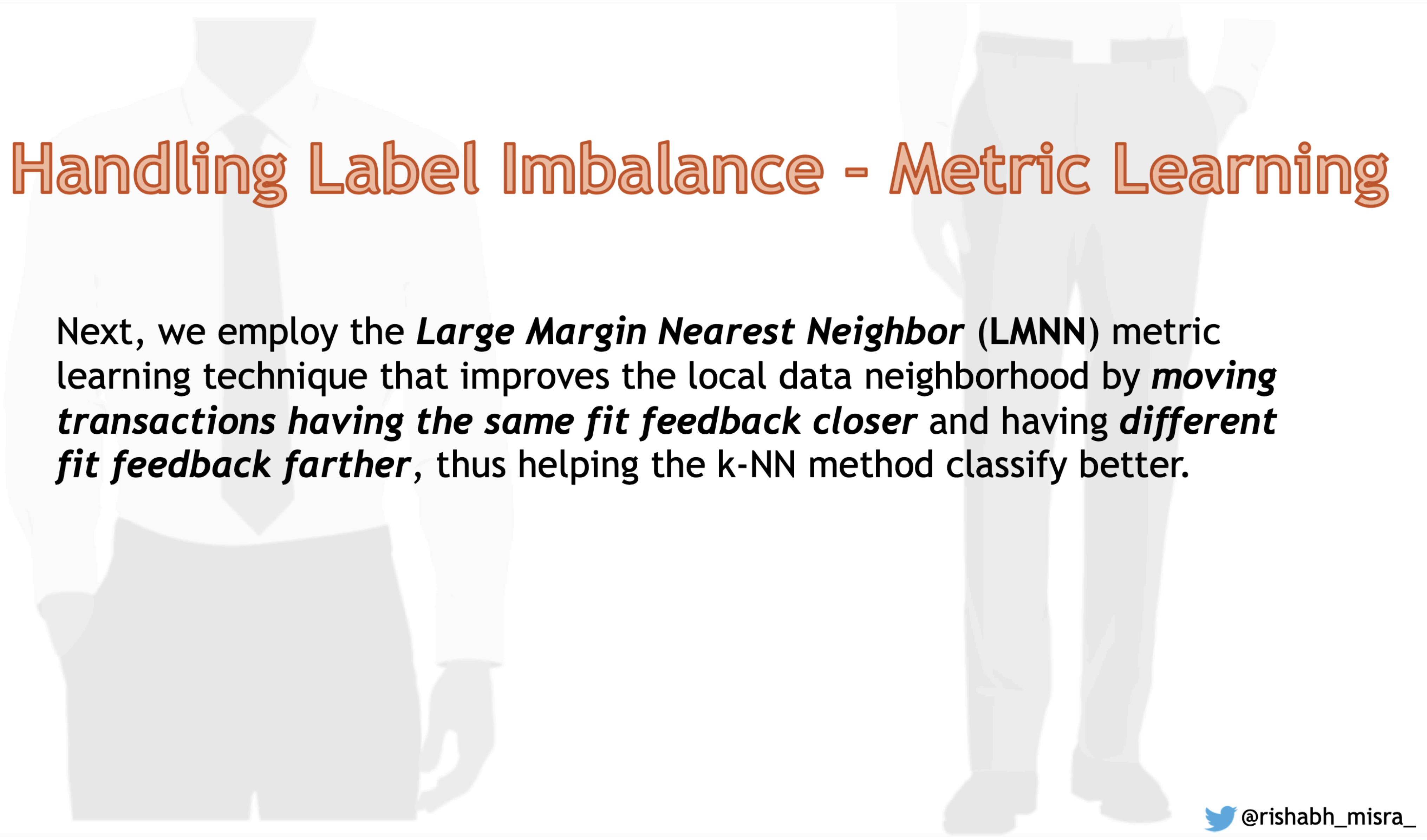
- This is similar to objective in Amazon's paper.
- We have just changed the definition of the *fit score* and added *monotonicity constraints*.
- We can optimize this objective using *Projected Gradient Descent* which is similar to *Stochastic Gradient Descent* with a difference that after every update the constraints are enforced.

Handling Label Imbalance - Prototyping

- The goal of prototyping techniques is to create certain number of "prototypes" from the available data such that they are representative of the whole data.
  - A prototype could be some key data sample from the dataset or could be a combination of several data samples. Usually, the number of prototypes created are way less than the number of data samples in the dataset.

- Our proposed prototyping technique first alters the training data distribution by strategically re-sampling from different classes.
  - Re-sampling is shown to be effective in handling label imbalance issues*.

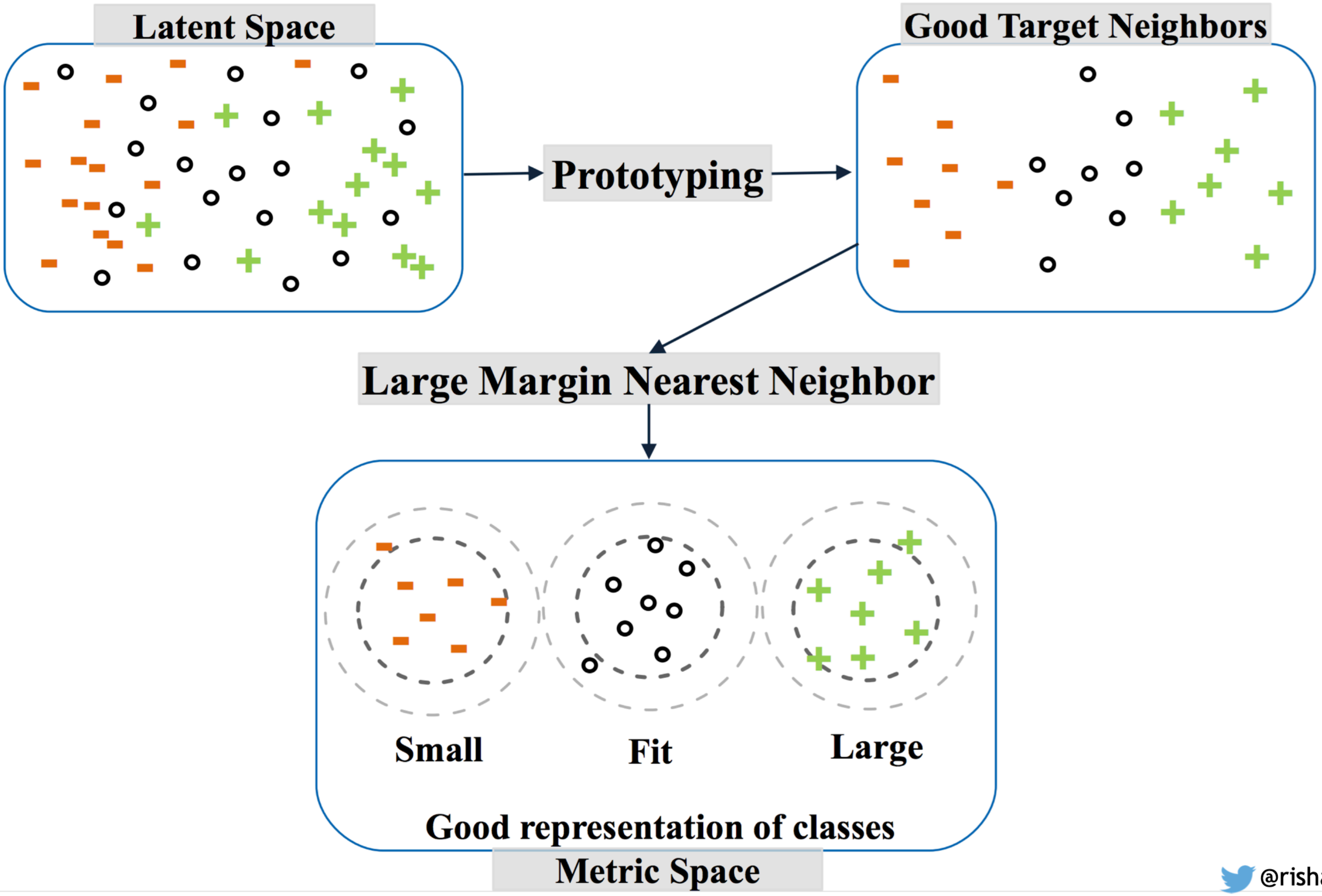
* Learning deep representation for imbalanced classification by Huang et al., In CVPR



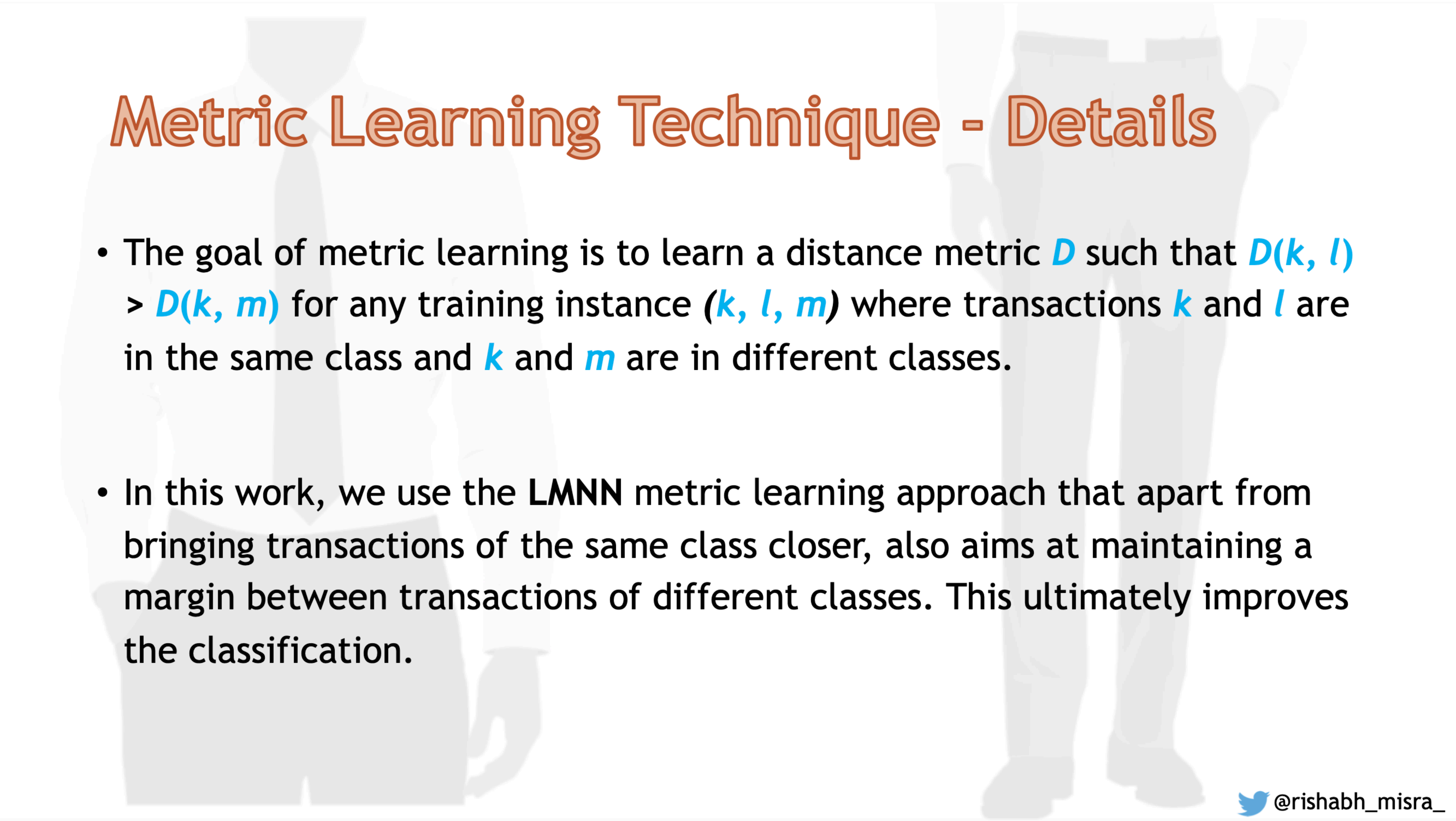
Handling Label Imbalance - Metric Learning

Next, we employ the ***Large Margin Nearest Neighbor*** (**LMNN**) metric learning technique that improves the local data neighborhood by ***moving transactions having the same fit feedback closer*** and having ***different fit feedback farther***, thus helping the k-NN method classify better.

@rishabh_misra_



@rishabh_misra_



Metric Learning Technique - Details

- The goal of metric learning is to learn a distance metric D such that $D(k, l) > D(k, m)$ for any training instance (k, l, m) where transactions k and l are in the same class and k and m are in different classes.

- In this work, we use the **LMNN** metric learning approach that apart from bringing transactions of the same class closer, also aims at maintaining a margin between transactions of different classes. This ultimately improves the classification.

@rishabh_misra_

Metric Learning Technique - Details

Specifically, **LMNN** does this by:

- Identifying the target neighbors for each transaction.
  - Target neighbors are those transactions that are *desired* to be closest to the transaction under consideration (that is, transactions from the same class).

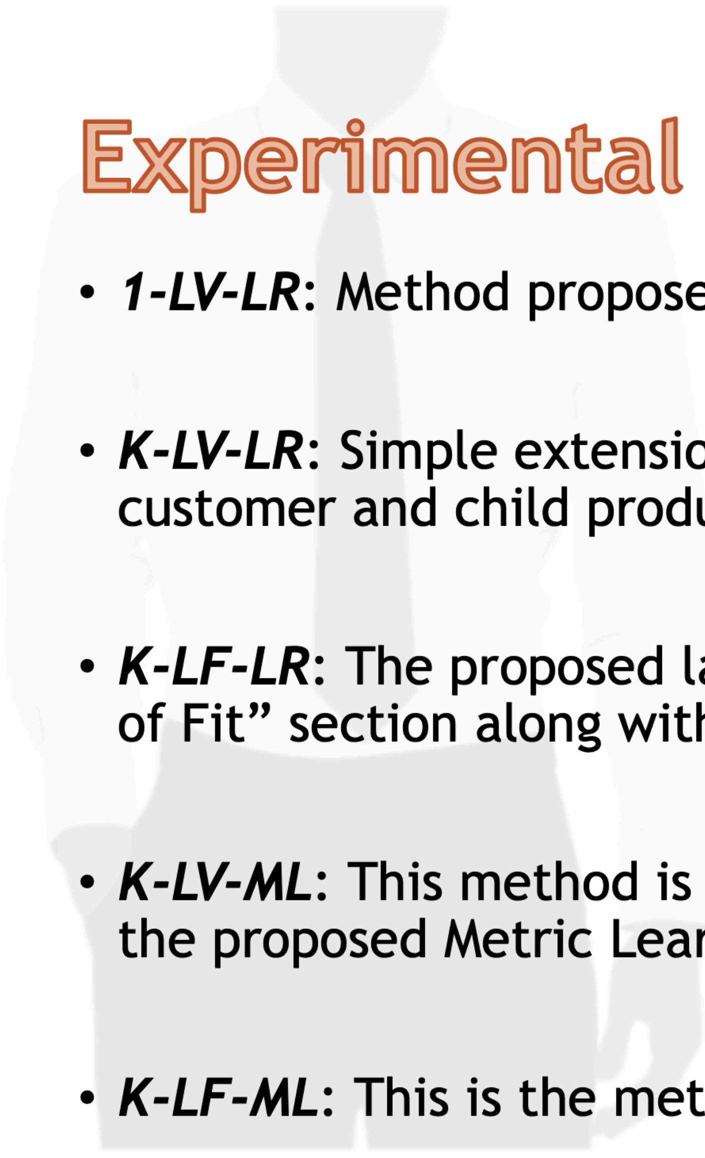
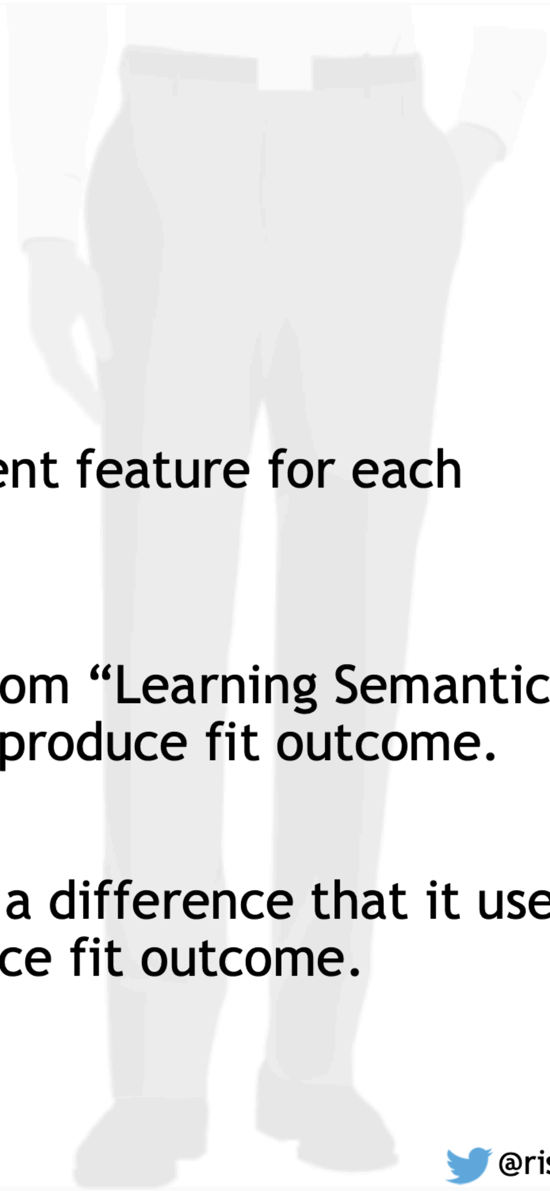
- Learning a linear transformation of the input space such that the resulting nearest neighbors of a transaction in the transformed space are indeed its target neighbors.
  - The final classification in **LMNN** is then given by applying **k**-NN in the transformed (metric) space. The distance measure ***D*** used by **LMNN** is the *Mahalanobis distance*.

@rishabh_misra_

Prototyping - Details

- One caveat in **LMNN** is that it fixes the **k** target neighbors for each transaction before it runs. This allows constraints to be defined locally.
- However, this also makes the method very sensitive to the ability of the Euclidean distance to select relevant target neighbors.
- To mitigate this limitation, we develop a heuristic that provides a good representation for each class by reducing noise from outliers and other non-contributing transactions (like the ones which are too close to the centroid of their respective class or to already selected transactions) by carefully sampling transactions.

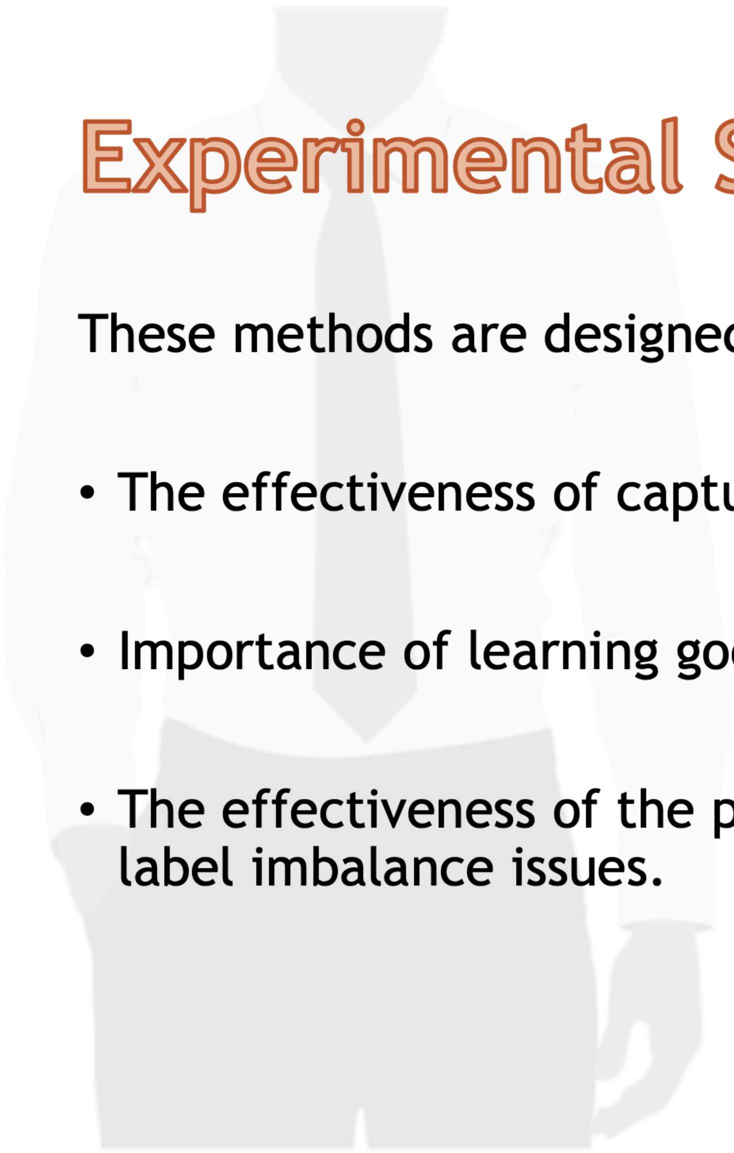
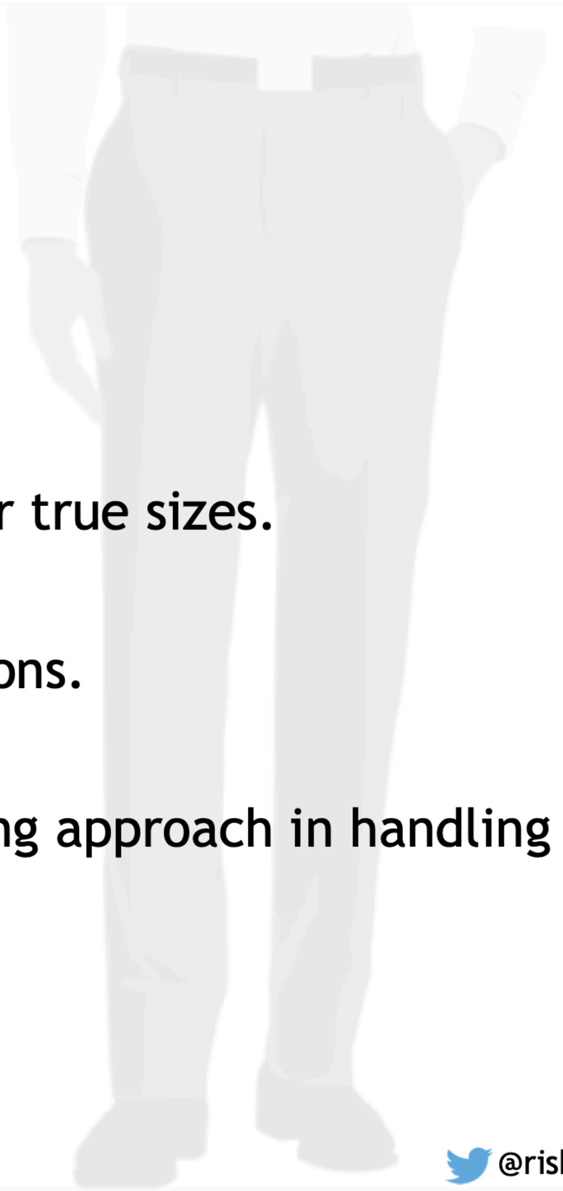
@rishabh_misra_

Experimental Setup

- ***1-LV-LR***: Method proposed by Amazon.
- ***K-LV-LR***: Simple extension of ***1-LV-LR*** where latent feature for each customer and child product is K dimensional.
- ***K-LF-LR***: The proposed latent factor variation from "Learning Semantics of Fit" section along with Logistic Regression to produce fit outcome.
- ***K-LV-ML***: This method is similar to ***K-LV-LR*** with a difference that it uses the proposed Metric Learning approach to produce fit outcome.
- ***K-LF-ML***: This is the method proposed by us.

@rishabh_misra_

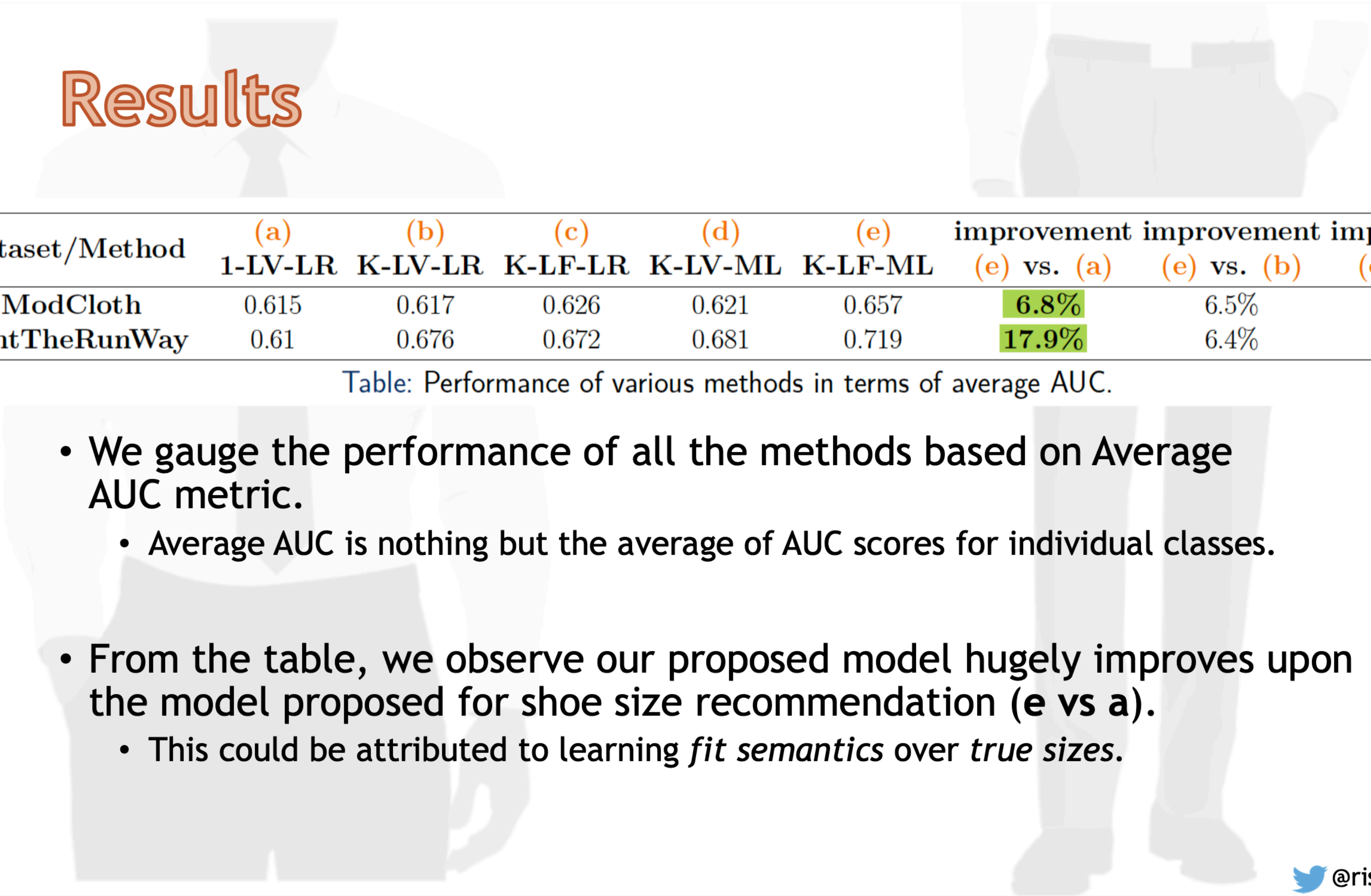



Experimental Setup

These methods are designed to evaluate:

- The effectiveness of capturing fit semantics over true sizes.
- Importance of learning good latent representations.
- The effectiveness of the proposed metric learning approach in handling label imbalance issues.

@rishabh_misra_



Results

Dataset/Method	(a) 1-LV-LR	(b) K-LV-LR	(c) K-LF-LR	(d) K-LV-ML	(e) K-LF-ML	improvement (e) vs. (a)	improvement (e) vs. (b)	improvement (e) vs (c)
ModCloth	0.615	0.617	0.626	0.621	0.657	**6.8%**	6.5%	4.9%
RentTheRunWay	0.61	0.676	0.672	0.681	0.719	**17.9%**	6.4%	7%

Table: Performance of various methods in terms of average AUC.

- We gauge the performance of all the methods based on Average AUC metric.
  - Average AUC is nothing but the average of AUC scores for individual classes.
- From the table, we observe our proposed model hugely improves upon the model proposed for shoe size recommendation (**e vs a**).
  - This could be attributed to learning *fit semantics* over *true sizes*.

Results

Dataset/Method	(a) 1-LV-LR	(b) K-LV-LR	(c) K-LF-LR	(d) K-LV-ML	(e) K-LF-ML	improvement (e) vs. (a)	improvement (e) vs. (b)	improvement (e) vs (c)
ModCloth	0.615	0.617	0.626	0.621	0.657	**6.8%**	6.5%	4.9%
RentTheRunWay	0.61	0.676	0.672	0.681	0.719	**17.9%**	6.4%	7%

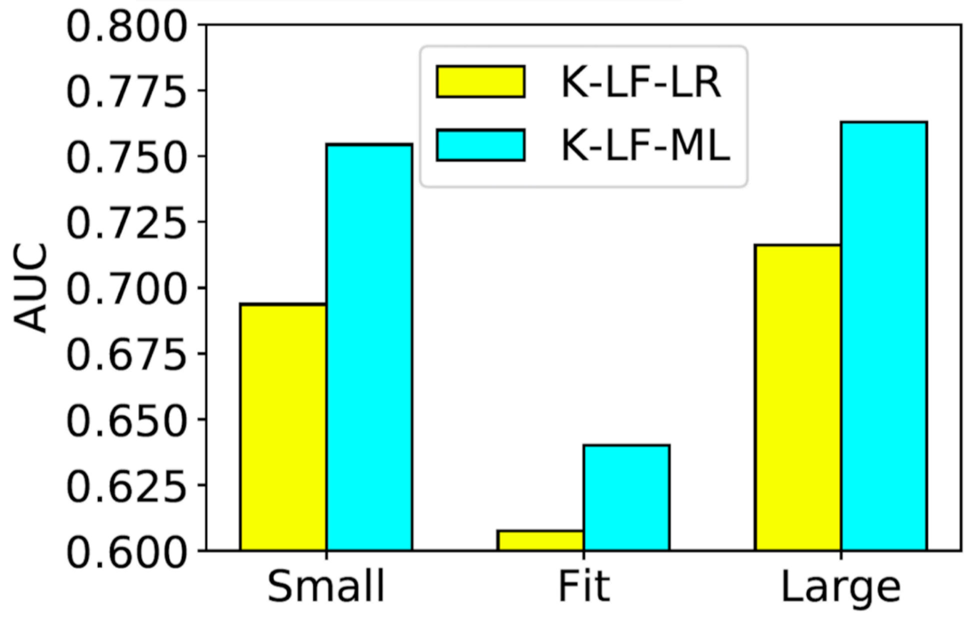
Table: Performance of various methods in terms of average AUC.

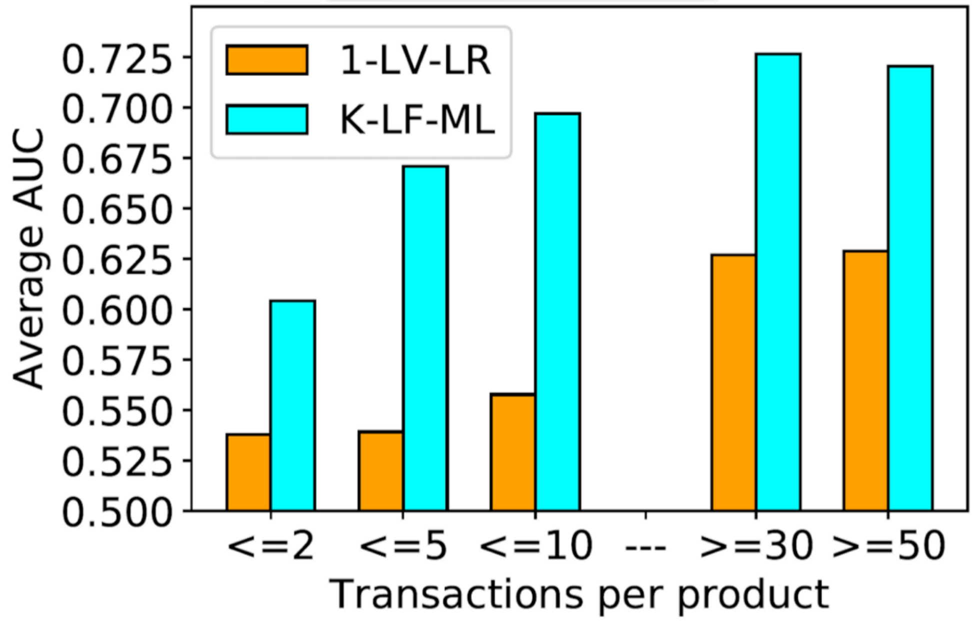
- We also observe that the improvements on ***ModCloth*** are relatively smaller than improvements on ***RentTheRunWay***.
  - This could be due to *ModCloth* having relatively more cold products and customers (products and customers with very few transactions) compared to *RentTheRunWay*.

- We also notice that metric learning approaches do not significantly improve performance when using representations from the ***K-LV*** method (**d vs b** and **e vs c**).
  - This underscores the importance of learning *good representations*.

@rishabh_misra_

Results

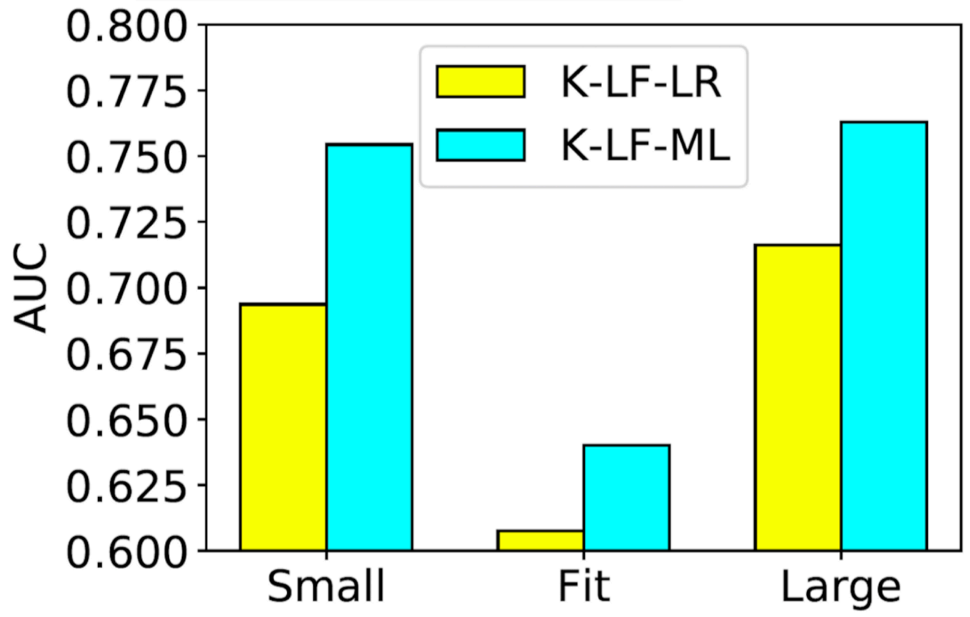


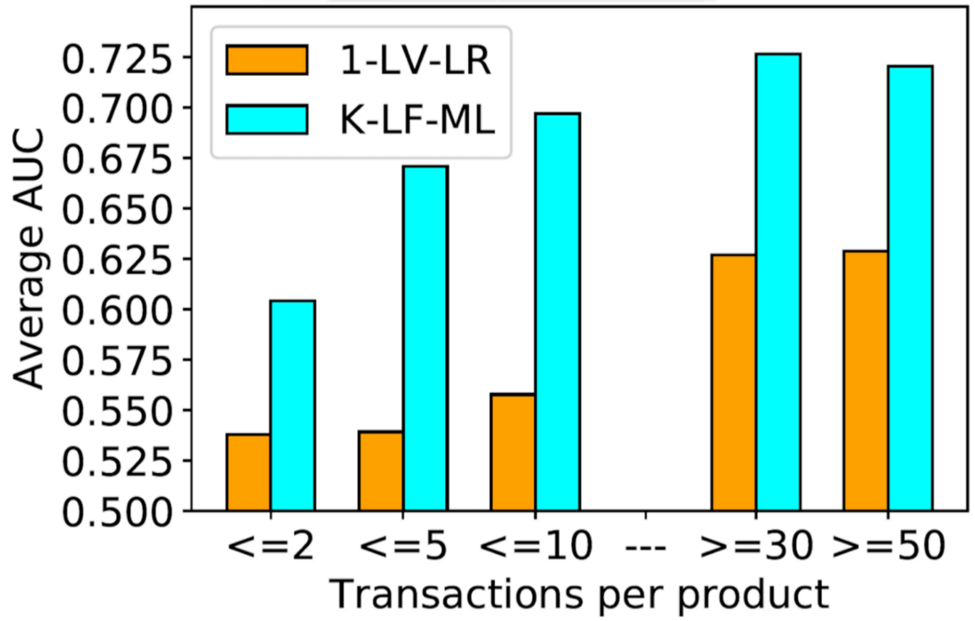


Besides learning good representations, good performance of ***K-LF-ML*** could also be ascribed to the ability of the proposed metric learning approach in handling label imbalance issues as depicted in the left-hand side graph above.

@rishabh_misra_

Results





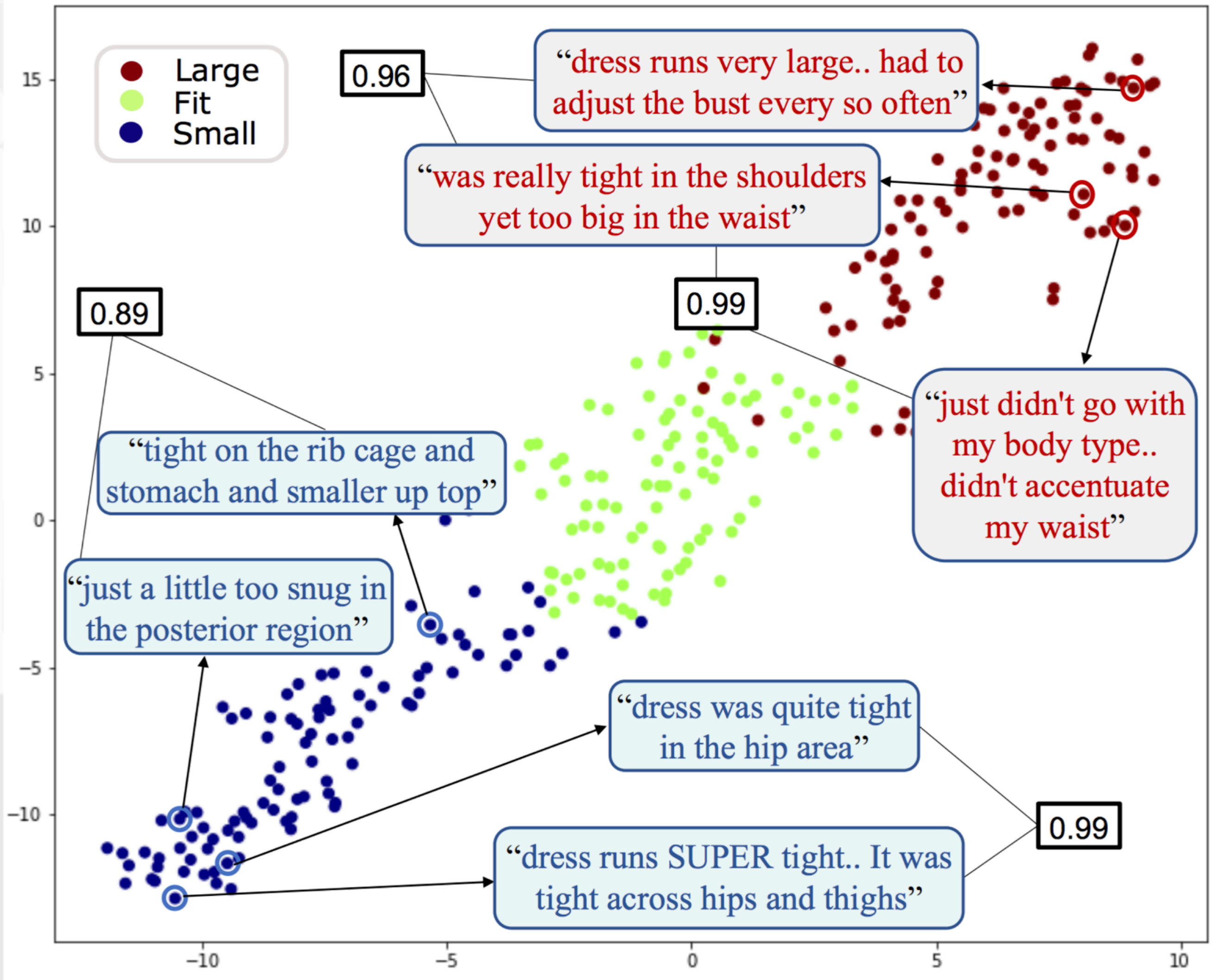
The right-hand side graph depicts how ***K-LF-ML*** performs in cold-start and warm-start scenarios. For cold products, we notice that ***K-LF-ML*** consistently performs better than ***1-LV-LR***, although their performances are slightly worse overall. As we consider products with more transactions, ***K-LF-ML*** improves quickly whereas the performance of ***1-LV-LR*** improves significantly only when sufficiently many samples are given.

@rishabh_misra_

Future Directions

- This is a very practical problem to solve with good implications and has several open directions.

- One such direction for further improvement could be to utilize the reviews to improve the interpretability of the model since currently, it is hard to understand what each latent dimension correspond to. This is possible by integrating a language model with the latent factor model to assign a specific meaning to each latent dimension (denoted by the corresponding topic) as done in this paper.



Qualitative Analysis of the learned latent features in Metric Space

@rishabh_misra_

Publication Resources

- Blog post on Towards Data Science : https://towardsdatascience.com/would-this-clothing-fit-me-5c3792b7a83f
- Research paper : https://cseweb.ucsd.edu/~jmcauley/pdfs/recsys18e.pdf
- GitHub repo with source code : https://github.com/rishabhmisra/Product-Catalog-Size-Recommendation-Framework
- Datasets : https://www.kaggle.com/rmisra/clothing-fit-dataset-for-size-recommendation

@rishabh_misra_

Thank you!

Questions?

@rishabh_misra_