Training Data Generation

Let's collect and label training data for the feed ranking ML model.

We'll cover the following

Your user engagement prediction model's performance will depend largely on the quality and quantity of the training data. So, let's see how you can generate training data for your model.

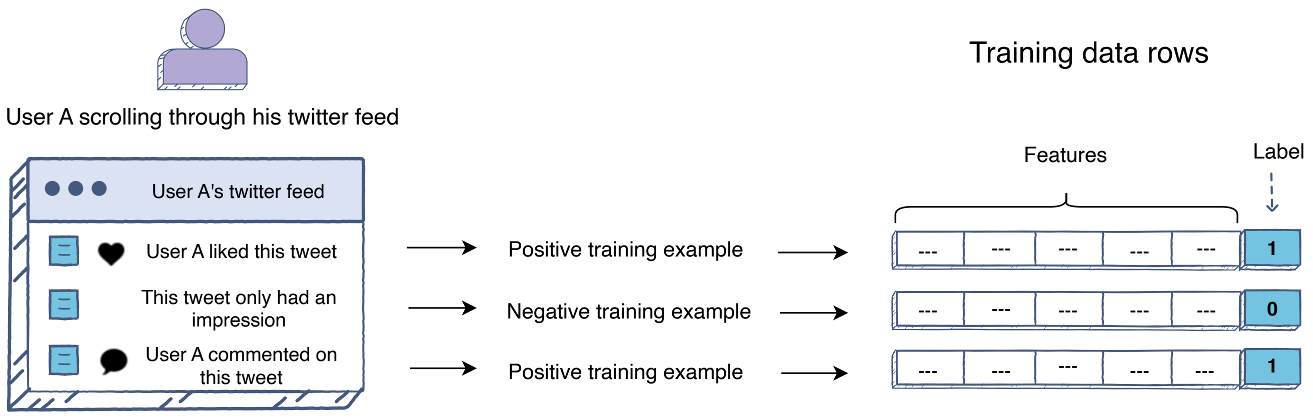
> 📝 Note that the term *training data row* and *training example* will be used interchangeably.

Training data generation through online user engagement#

The users' online engagement with Tweets can give us positive and negative training examples. For instance, if you are training a single model to predict user engagement, then all the Tweets that received user engagement would be labeled as positive training examples. Similarly, the Tweets that only have impressions would be labeled as negative training examples.

> 📝 **Impression**: If a Tweet is displayed on a user's Twitter feed, it counts as an impression. It is not necessary that the user reads it or engages with it, scrolling past it also counts as an impression.

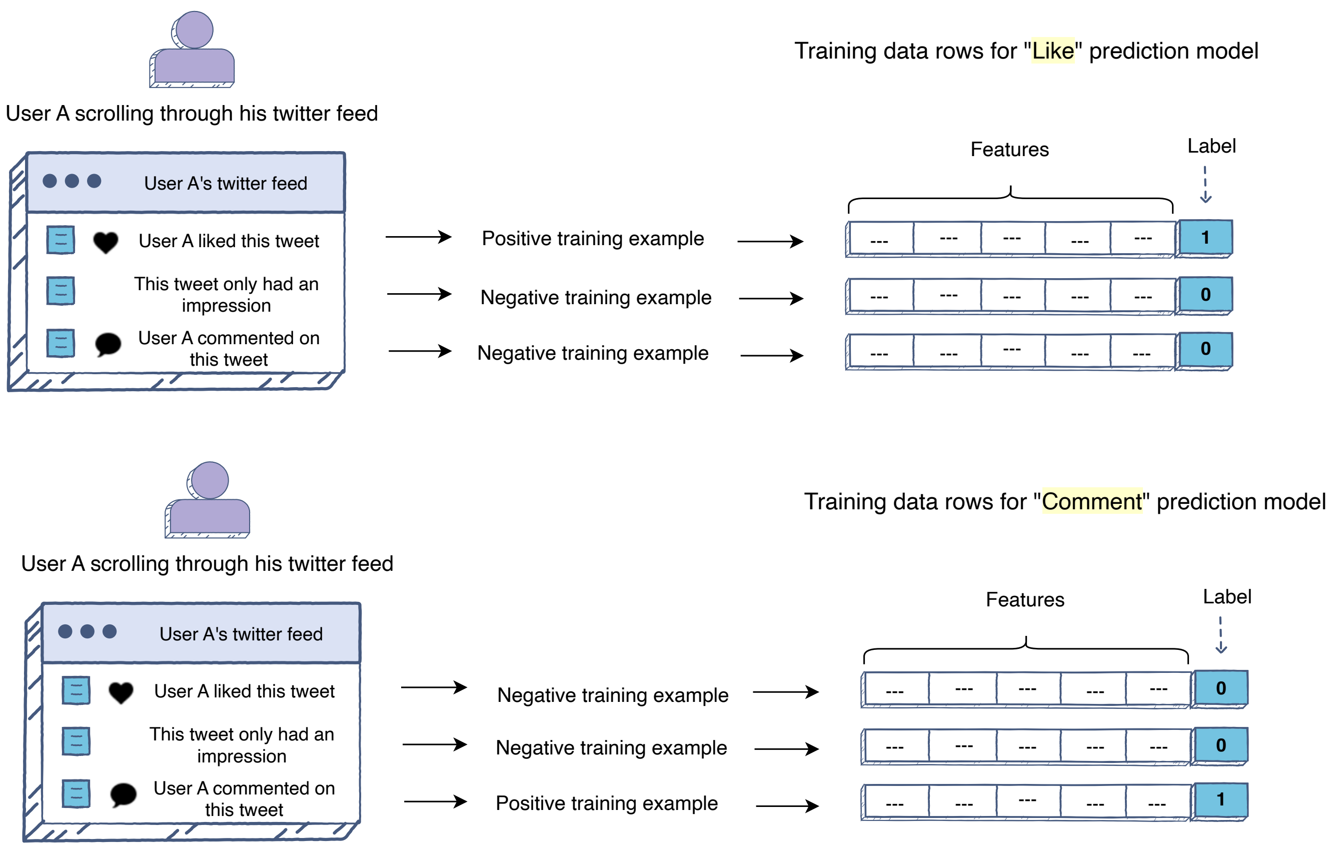
Training data generation for a single model to predict user engagement


Training data rows
User A scrolling through his twitter feed
User A's twitter feed
User A liked this tweet
This tweet only had an impression
User A commented on this tweet
Positive training example
Negative training example
Positive training example
Features
Label
--- --- --- --- --- 1
--- --- --- --- --- 0
--- --- --- --- --- 1

Any user engagement counts as positive example

However, as you saw in the architectural components lesson, that you can train different models, each to predict the probability of occurrence of different user actions on a tweet. The following illustration shows how the same user engagement (as above) can be used to generate training data for separate engagement prediction models.

Training data generation for separate models to predict user engagement


Training data rows for "Like" prediction model
User A scrolling through his twitter feed
User A's twitter feed
User A liked this tweet
This tweet only had an impression
User A commented on this tweet
Positive training example
Negative training example
Negative training example
Features
Label
1
0
0
Training data rows for "Comment" prediction model
User A scrolling through his twitter feed
User A's twitter feed
User A liked this tweet
This tweet only had an impression
User A commented on this tweet
Negative training example
Negative training example
Positive training example
Features
Label
0
0
1

When you generate data for the "Like" prediction model, all Tweets that the user has liked would be positive examples, and all the Tweets that they did not like would be negative examples.

> 📝 Note how the comment is still a negative example for the "Like" prediction model.

Similarly, for the "Comment" prediction model, all Tweets that the user commented on would be positive examples, and all the ones they did not comment on would be negative examples.

Balancing positive and negative

training examples#

Models essentially learn behavior from the data we present them with. Therefore, it's important for us to provide a good sample of both positive and negative examples to model these interactions between different actors in a system. In the feed-based system scenario, on average, a user engages with as little as approximately 5% of the Tweets that they view per day. How would this percentage affect the ratio of positive and negative examples on a larger scale? How can this be balanced?

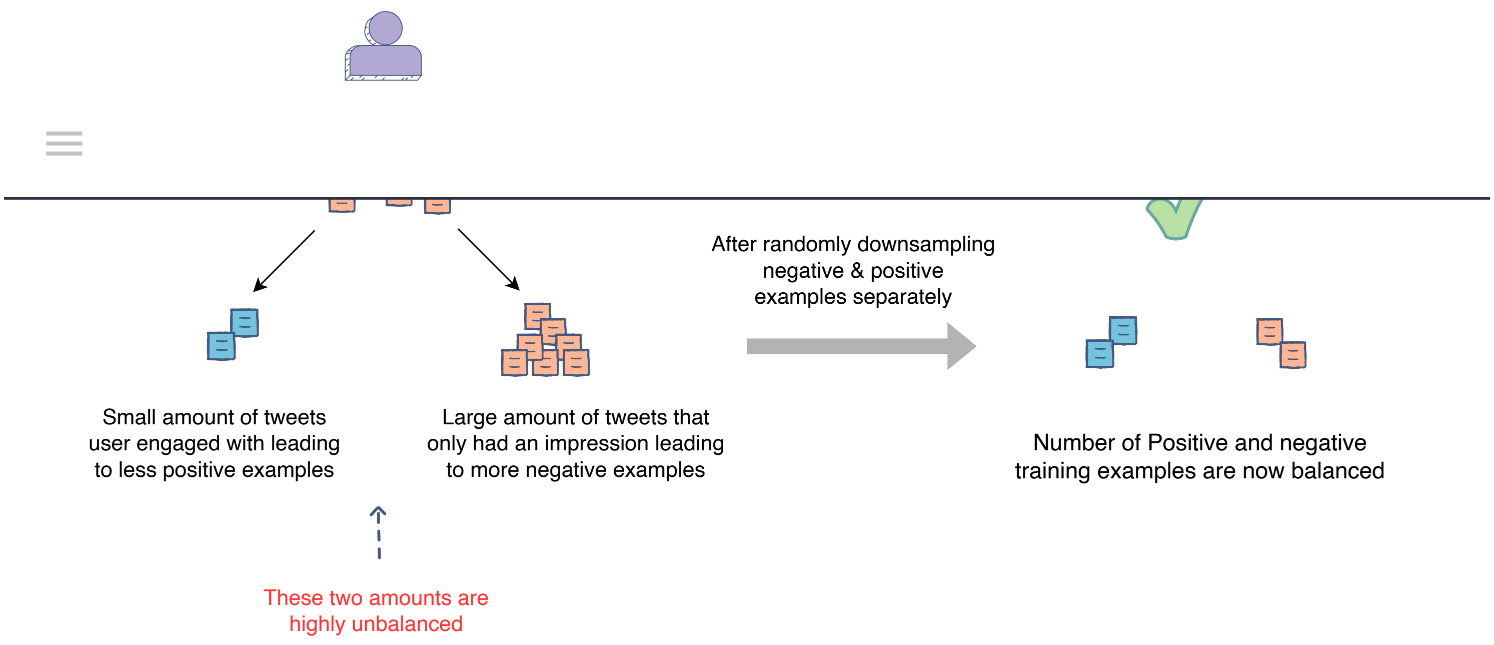
Looking at the bigger picture, assume that one-hundred million Tweets are viewed collectively by the users in a day. With the 5% engagement ratio, you would be able to generate five million positive examples, whereas the remaining ninety-five million would be negative.

Let's assume that you want to limit your training samples to ten million, given that the training time and cost increases as training data volume increases. If you don't do any balancing of training data and just randomly select ten million training samples, your models will see only 0.5 million positive examples and 9.5 million negative examples. In this scenario, the models might not be able to learn *key* positive interactions.

Therefore, in order to balance the ratio of positive and negative training samples, you can **randomly downsample**:

- negative examples to five million samples
- positive examples to five million samples

Now, you would have a total of **ten million training examples per day**; five million of which are positive and five million are negative.


After randomly downsampling negative & positive examples separately
Small amount of tweets user engaged with leading to less positive examples
Large amount of tweets that only had an impression leading to more negative examples
Number of Positive and negative training examples are now balanced
These two amounts are highly unbalanced

📝 **Note**

If a model is *well-calibrated*, the distribution of its predicted probability is similar to the distribution of probability observed in the training data. However, as we have changed the sampling of training data, our model output scores will **not** be well-calibrated. For example, for a tweet that got only 5% engagement, the model might predict 50% engagement. This would happen because the model is trained on data that has an equal quantity of negative and positive samples. So the model would think that it is as likely for a tweet to get engagement as it is to be ignored. However, we know from the training that this is not the case.

Given that the model's scores are only going to be used to rank Tweets among themselves, poor model calibration doesn't matter much in this scenario. We will discuss this in the ads system chapter, where calibrated scores are important, and we need to be mindful of such a sampling technique.

Train test split#

Train test split#

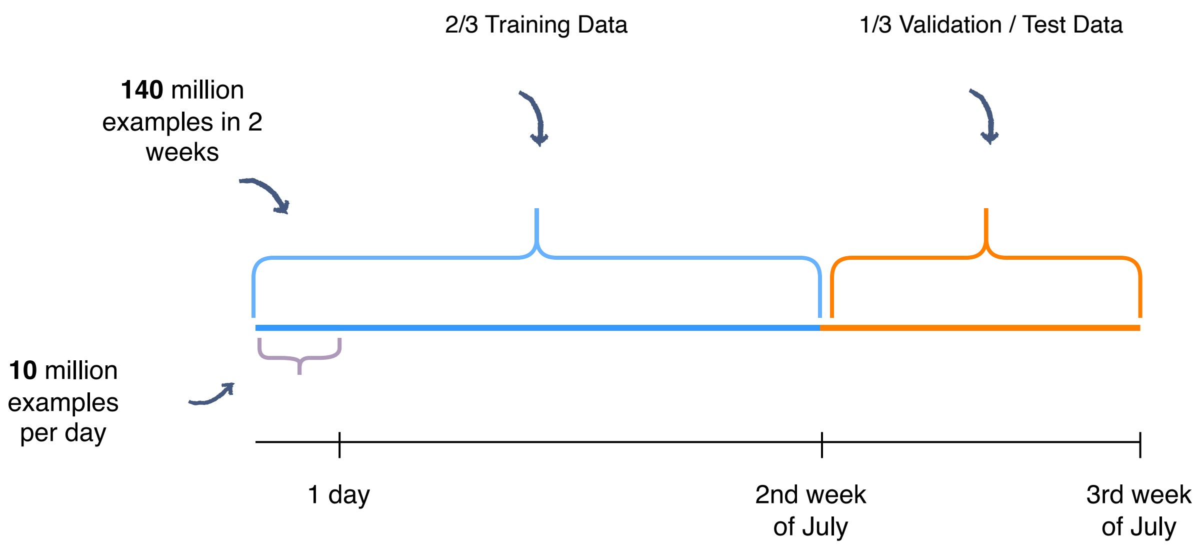
You need to be mindful of the fact that the user engagement patterns may differ throughout the week. Hence, you will use a week's engagement to capture all the patterns during training data generation. At this rate, you would end up with around seventy million rows of training data.

You may randomly select $\frac{2}{3}^{rd}$, or 66.6%, of the seventy million training data rows that you have generated and utilize them for training purposes. The rest of the $\frac{1}{3}^{rd}$, or 33.3%, can be used for validation and testing of the model.

However, this random splitting defeats the purpose of training the model on an entire week's data. Also, the data has a time dimension, i.e., we know the engagement on previous Tweets, and we want to predict the engagement on future Tweets ahead of time. Therefore, you will train the model on data from one time interval and validate it on the data with the succeeding time interval. This will give a more accurate picture of how the model will perform in a real scenario.

> 📝 We are building models with the intent to forecast the future.

In the following illustration, we are training the model using data generated from the first and second week of July and data generated in the thirdweek of July for validation and testing purposes.


2/3 Training Data
1/3 Validation / Test Data
140 million examples in 2 weeks
10 million examples per day
1 day
2nd week of July
3rd week of July

Splitting data for training, validation, and testing

> 📝 Have a look at the training data collection strategies lesson to get more insights.

Back

Feature Engineering

Next

Ranking

✓ Mark as Completed

Report an Issue