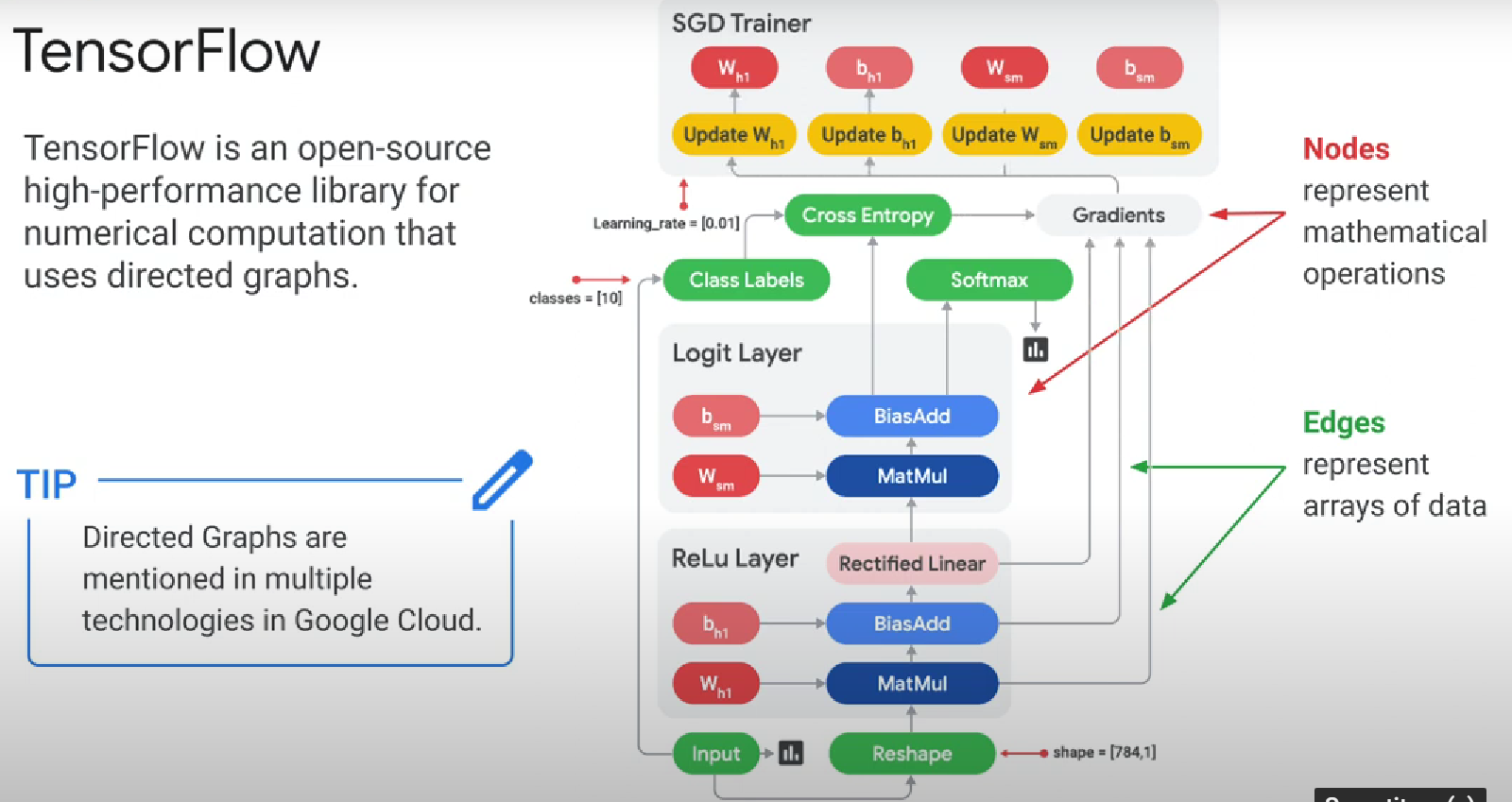


Natural Language API:

\* score of the sentiment ranges between -1.0 (negative) and 1.0 (positive) and corresponds to the overall emotional leaning of the text.

\* magnitude indicates the overall strength of emotion (both positive and negative) within the given text, between 0.0 and +inf. Unlike score, magnitude is not normalized; each expression of emotion within the text (both positive and negative) contributes to the text's magnitude (so longer text blocks may have greater magnitudes).



TPUs: tensorflow processing units

Steps to build an ML model:

**(NOTE: AI PLATEFORM HAS BEEN REPLACED BY VERTEX AI)**

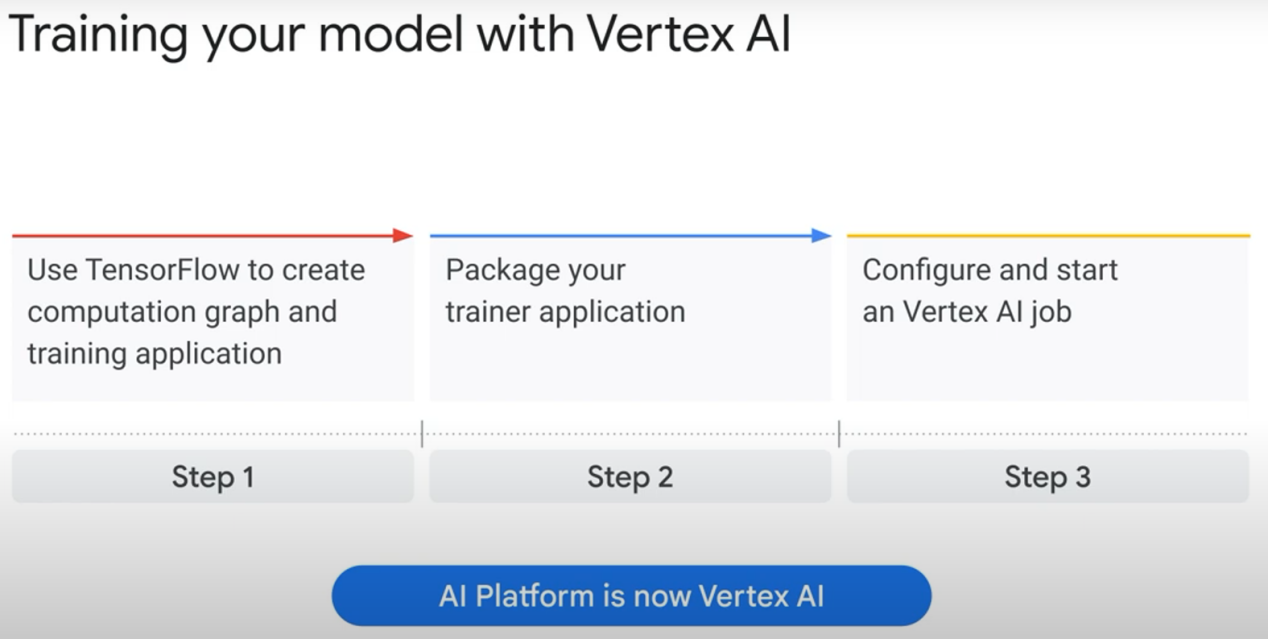
\* Use a **sample of the data** and train a model (you can do this in a **notebook**)

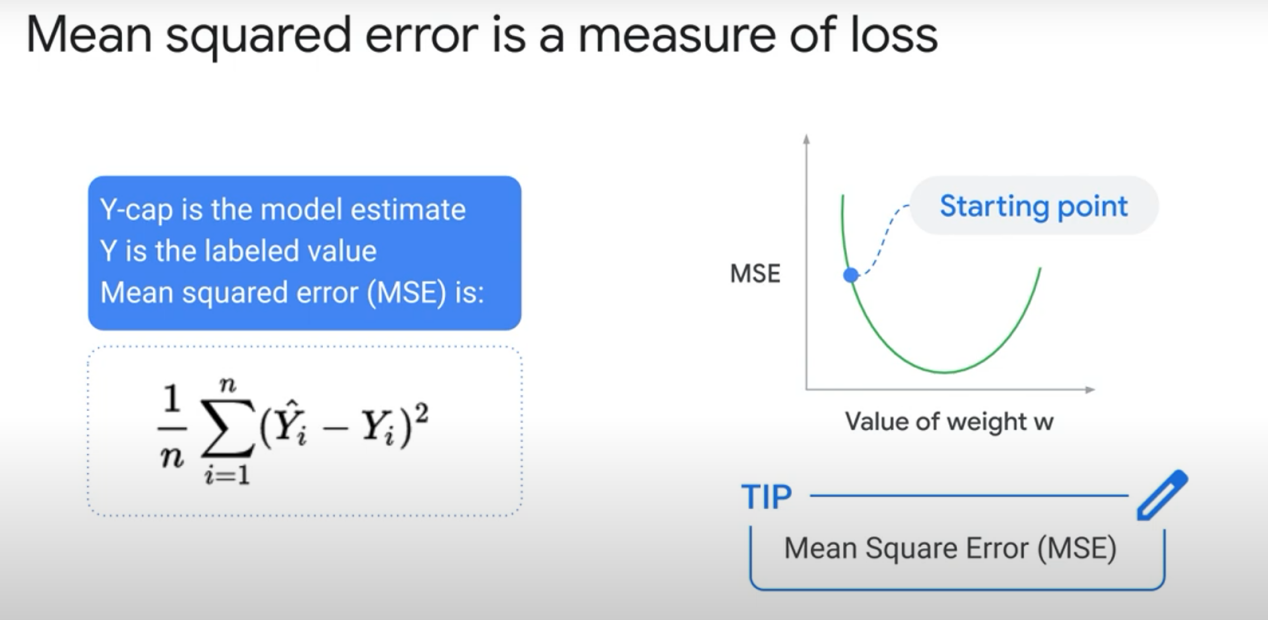
\* Once you’re satisfied, scale your model to google cloud:

1. Store your model in Cloud Storage
2. You can use the «gcloud ai-platform local predict» command to test how your model serves predictions before you deploy it to AI Platform Prediction.
3. AI Platform Prediction organizes your trained models using model and version resources. An AI Platform Prediction model is a container for the versions of your machine learning model. Create a model in the AI plateform and link it to your stored model in cloud storage.

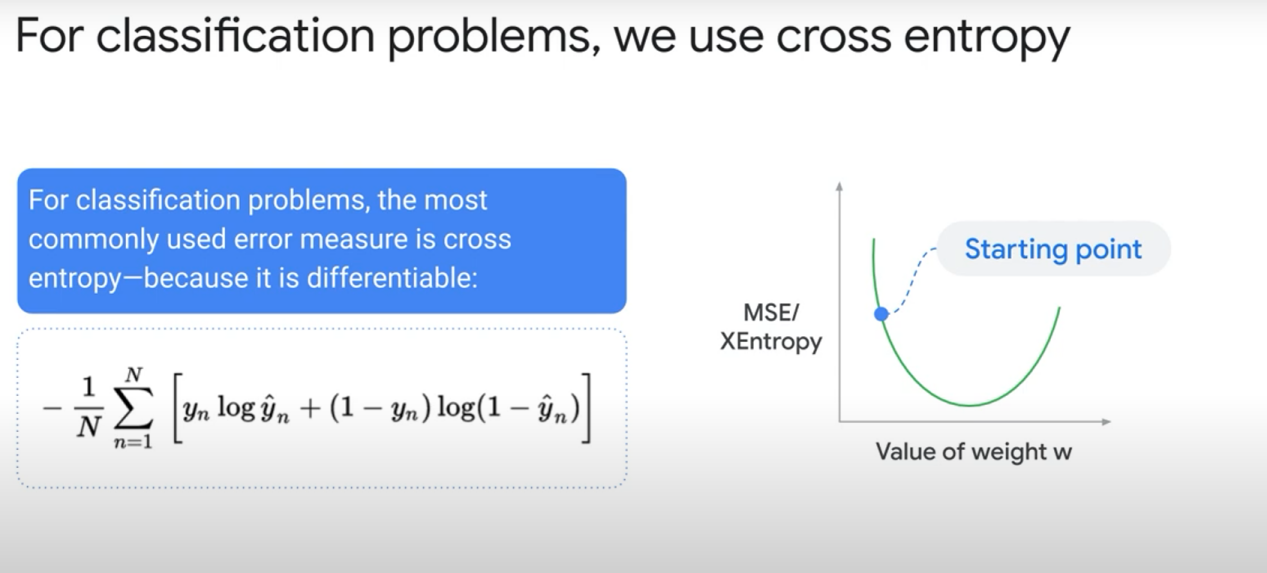
OR

1. (for vertex AI) gcloud beta ai models upload (to upload the model in a prebuilt vertex container)
2. Create an Endpoint (gcloud ai endpoints create) and deploy your model to that endpoint (gcloud ai endpoints deploy-model )

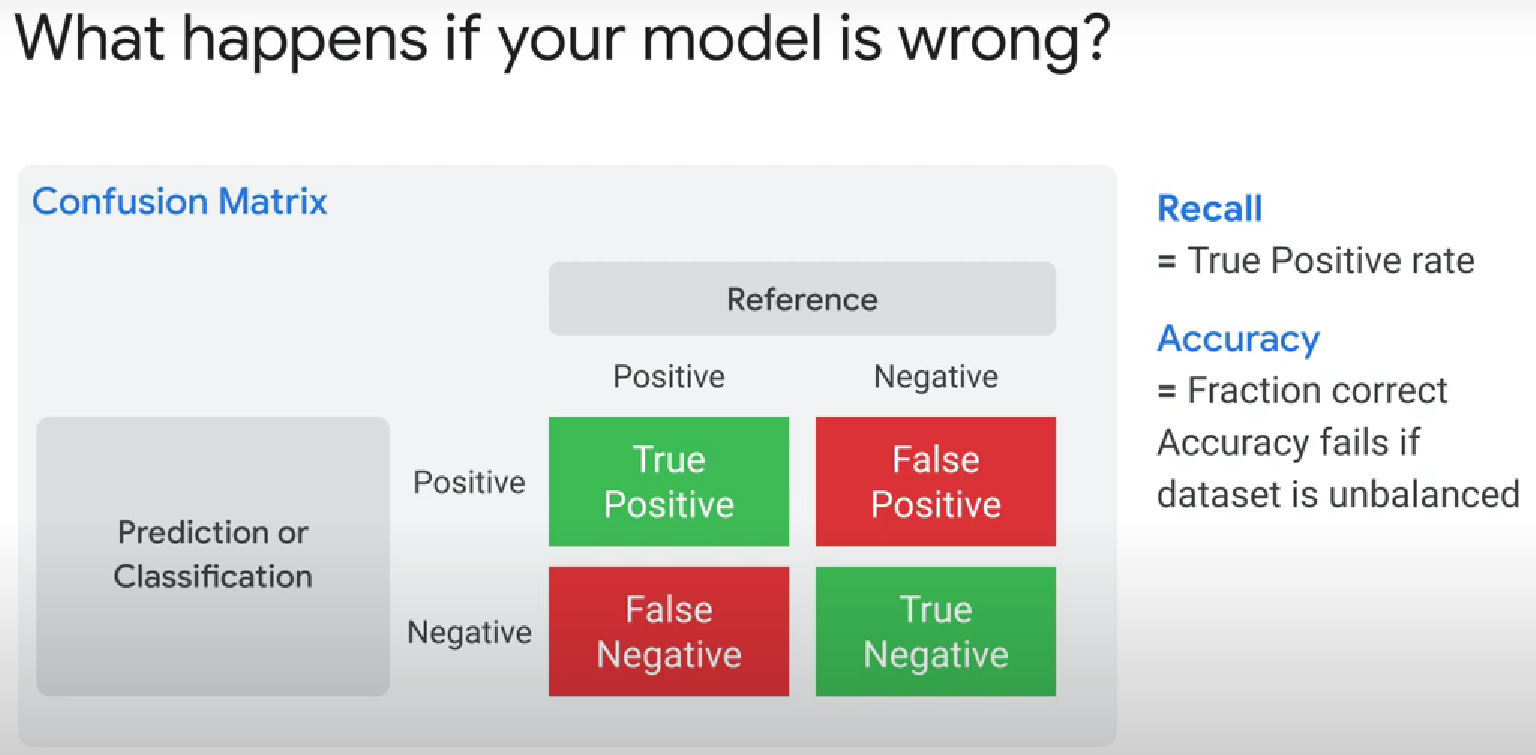




Note RMSE = root of MSE, it is easier to use it as the remc is in the units of the measurement making it and understand the significance of the value



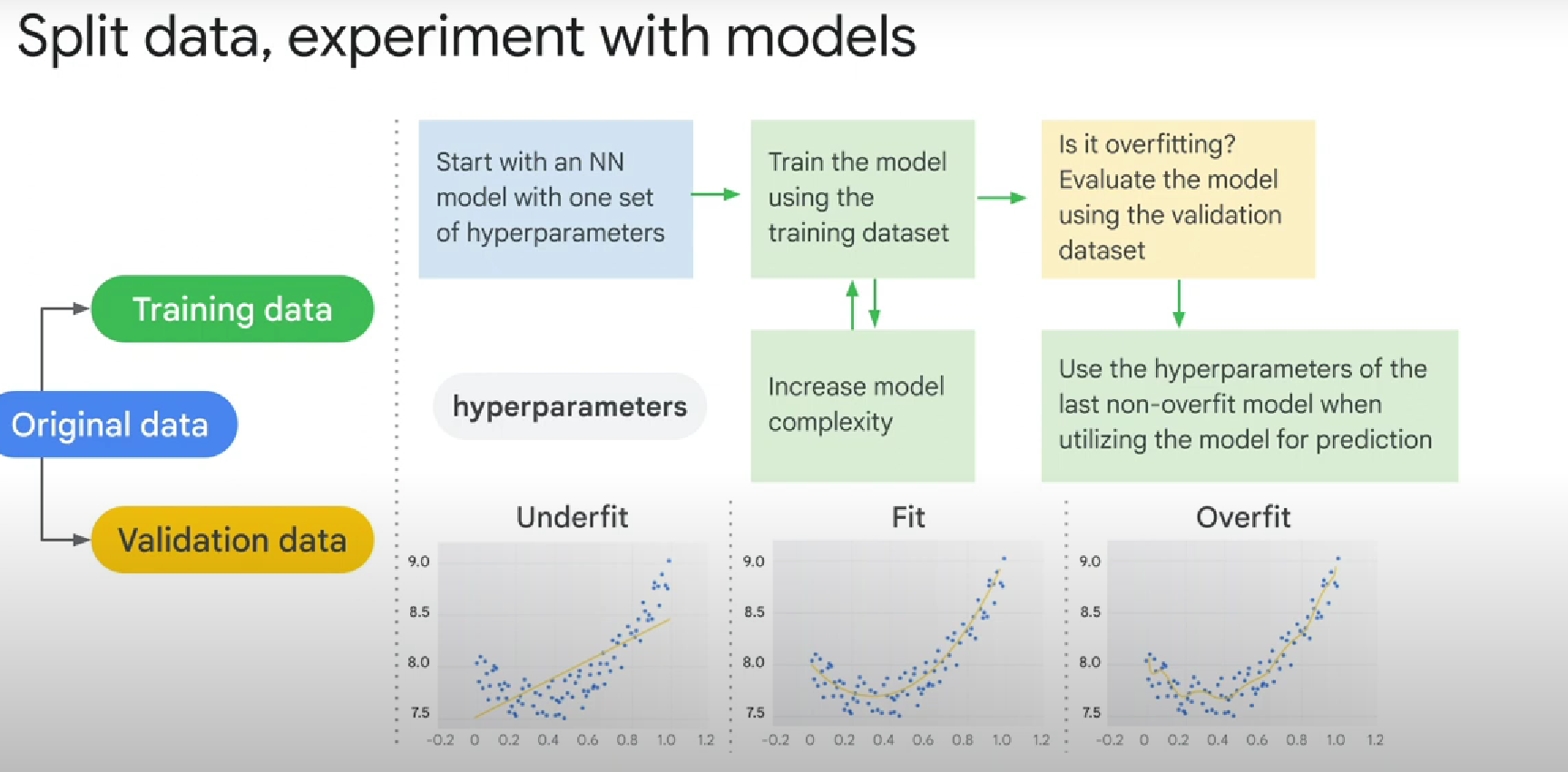
If the question describes **Cross entropy** = **Classification** ML problem



Use confusion matrixes to describe the performance of a ML model



Note: categorizing produces **discrete values** and regression produces **continuous values**

if someone said they had a machine learning model that recognizes new instances and categorizes them correctly 100 of the time == it would be an indicator that the validation data somehow got mixed up with the training data and that the data is no longer a good measure of how well the model's (Overfitting)

Must-know:

**Data scarcity** is when a) there is limited amount or a complete lack of labeled training data, or b) lack of data for a given label compared to the other labels

if the question says data is scarce then you should be thinking independent test data or cross-validate our candidate answers

**Cross-validation** is a resampling method that uses different portions of the data to test and train a model on different iterations.

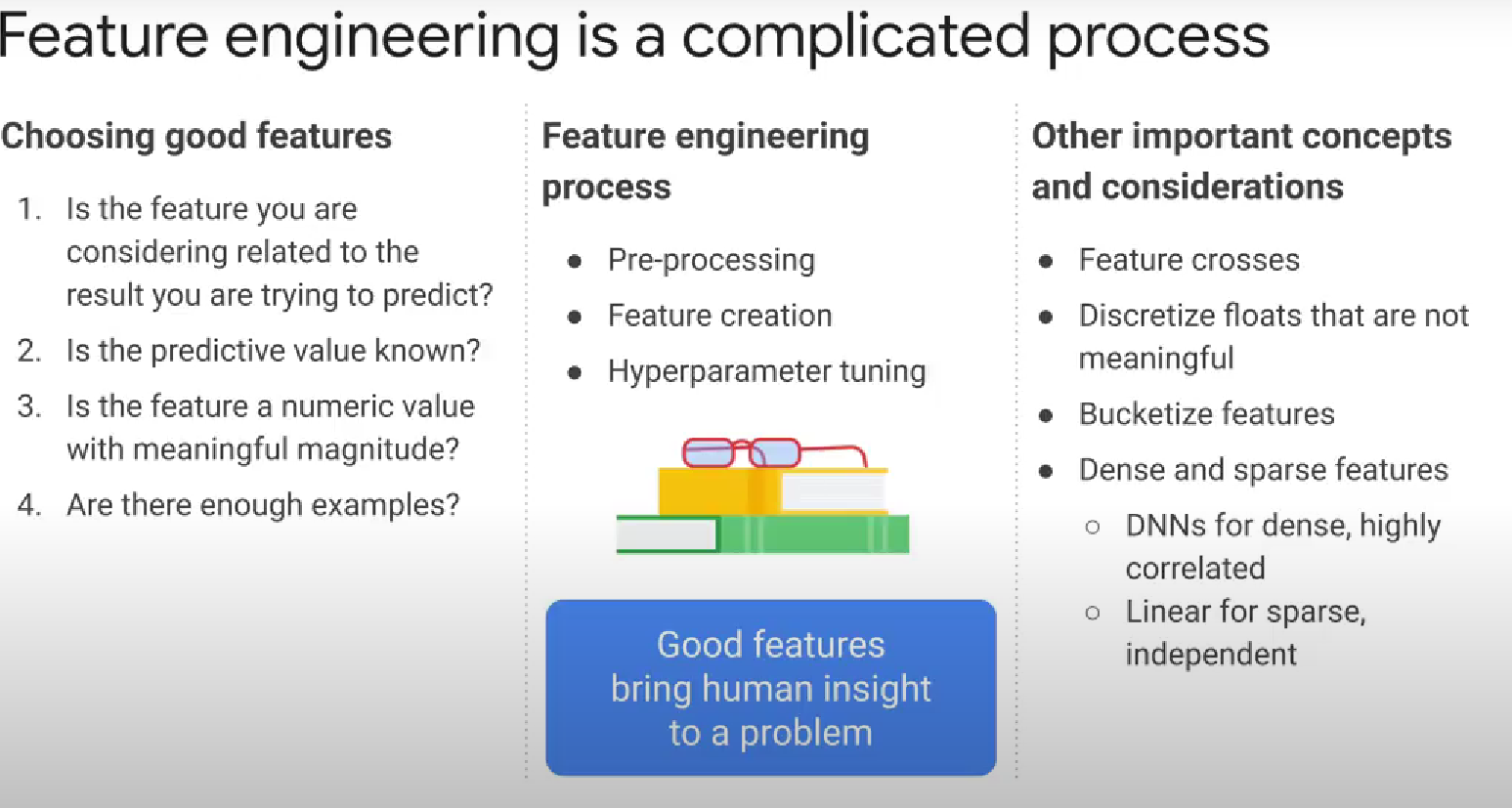
Basically, if we do not have too much data (data scarcity), we don’t want to loose 10% of our dataset for validation (80% training data, 10 validation, 10 testing). We do 90% training and 10% testing.

For each iteration, we split our training data to k «folds» or groups. We train the model on k-1, and validate on the remaining fold.

Tip: **Edge computing** == **model processing is pushed closer to the inputs**.

For example, doing Machine Learning processing **closer to the IoT sensors**, by performing work in nearby datacenters or regions, is 'edge computing'

Tip: One common **source of error** is accidental inclusion of **biased data** in the data being used for model training or validation.

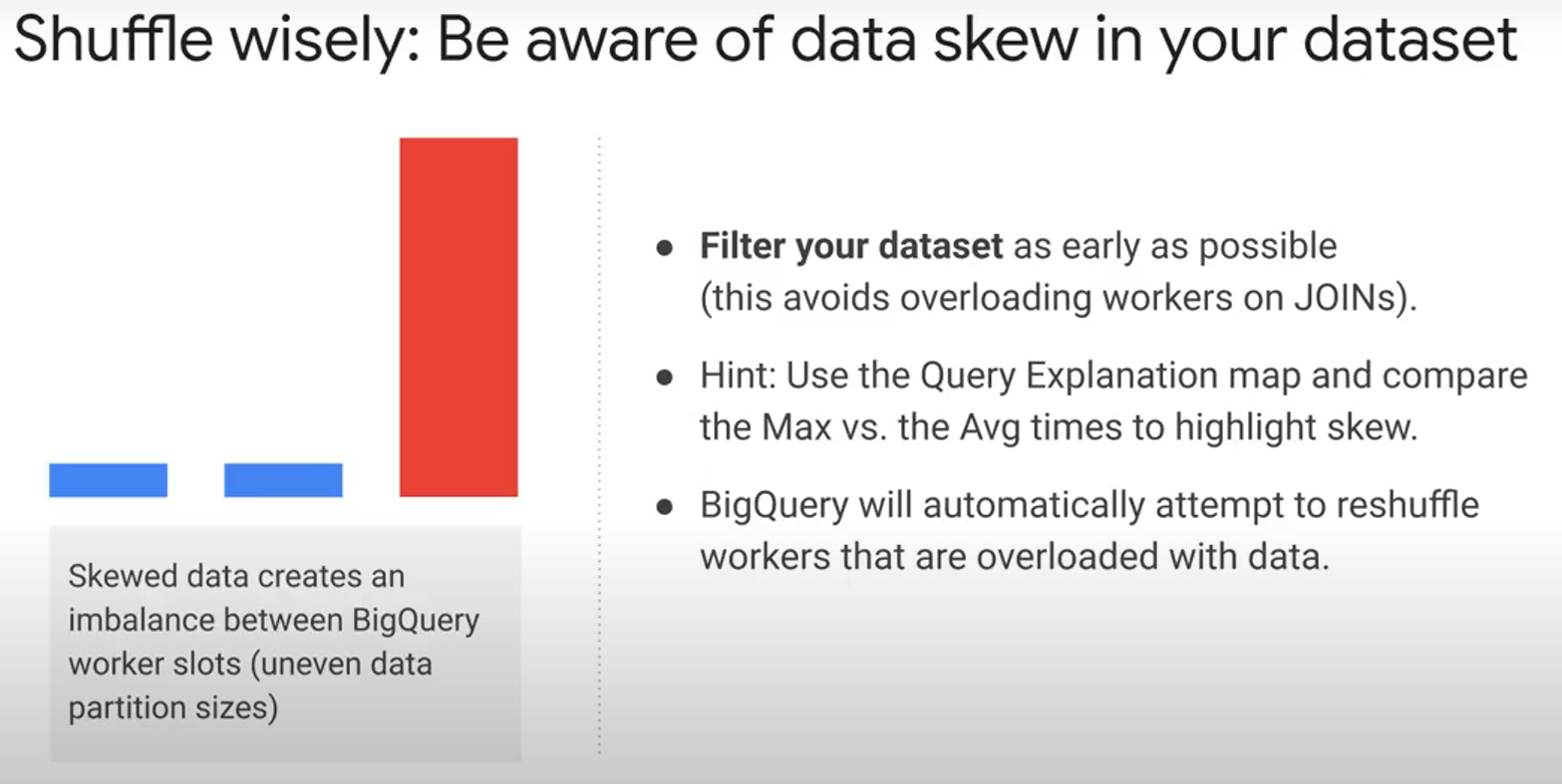


**Bucketizing**: putting float values into categories.

Features with **sparse data** are features that have mostly zero values. => linear model

**A feature cross** is a synthetic feature formed by multiplying (crossing) two or more features.

**Learning-rate (step size)**



Big difference between average expected excution time of a query and the max time == data skew( uneven partitions) , very probably