Machine Learning Data Lifecycle in Production

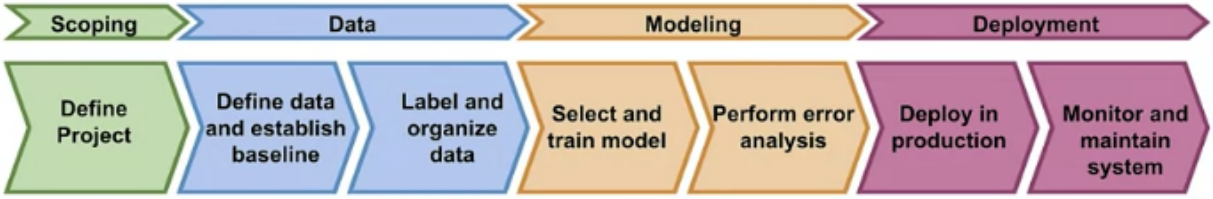
Week 1 - Collecting, Labelling and Validating Data

This week they talked about Machine Learning Engineering in Production, Collecting Data, Labeling Data and Validating Data.

Machine Learning Engineering in Production

A production machine learning system can be seen like the image below and listed here:

- Scoping: define the project to be worked on
- Data: define the data to be used and establish the baseline to surpass/reach, label and organize the data available
- Modeling: select and train a model, perform error analysis on the model trained
- Deployment: deploy the model in production, monitor and maintain the system afterwards



The suggestion was to use TFX (TensorFlow Extended), which is an open-source end-to-end ML platform.

Collecting Data

They elaborate on some points, such as: dataset issues (inconsistent formatting, compounding errors from other ML models, monitor data sources for system issues and outages), translate user needs into data problems, ensure data coverage and high predictive signal, and also source, store and monitor quality data responsibly.

Labeling Data

They elaborate two major points:

- Direct labeling: continuous creation of training dataset (default in credit application, e.g.)
- Human labeling: people ("raters") to examine data and assign labels manually

Validating Data

They mostly focused on detecting data issues, separating the concepts within two properties:

- Drift: changes in data over time
- Skew: difference between two static versions (such as training and serving set)

The issues can be:

- **Data drift:** a change in statistical property of the features (changes in the world or external factors, e.g.)
- **Concept drift:** a change in statistical property of the labels over time (somehow the meaning and relation of the variables and the target changed)
- **Schema skew:** training and serving data do not conform to the same schema (expecting string but getting an integer, e.g.)
- **Distribution skew:** is a divergence in training and serving dataset in terms of covariance

Below there is a snippet for checking schema skew using TensorFlow Data Validation (tfdv).

```
# Infer the data schema by using the training statistics that you generated
schema = tfdv.infer_schema(statistics=train_stats)

# Calculate and display anomalies.
anomalies = tfdv.validate_statistics(statistics, schema)
```

Week 2 - Feature engineering, transformation and selection

This week they talked about different methodologies for feature engineering, feature transformation at scale and feature selection.

Feature Engineering

- Data preprocessing: transforms raw data into a clean and training-ready dataset
- Feature engineering maps:
  - Raw data into feature vectors
  - Integer values to floating-point values
  - Normalizes numerical values
  - Strings and categorical values to vectors of numeric values
  - Data from one space into a different space

Some feature engineering techniques are:

- **Feature scaling:** converts values from their natural range into a prescribed range (e.g., image pixel scale is [0, 255] rescaled to [-1, 1])
- **Normalization:** also called the min-max normalization ((X-X_min)/(X_max-X_min)) specially used for data that you know it's not Gaussian (follow the Normal distribution)
- **Standardization:** or z-score, relates the number of standard deviations away from the mean ((X-mu)/std) ~ N(0, std)
- **Bucketizing / Binning:** creating categories out of intervals or grouping existing categories
- **Other techniques:** some other techniques are for dimensionality reduction in embeddings, such as principal component analysis (PCA) and t-Distributed stochastic neighbor embedding (t-SNE).

Feature Transformation at Scale

They talk about preprocessing data at scale and the challenges faced. For example, different ways of performing feature transforming (listed below).

Pre-processing training dataset

Pros	Cons
Run-once	Transformations reproduced at serving
Compute on entire dataset	Slower iterations

Transforming within the model

Pros	Cons
Easy iterations	Expensive transforms
Transformation guarantees	Long model latency
	Transformations per batch: skew

It can be done using TensorFlow Transform, snippet below.

```
def preprocessing_fn(inputs):
    """Preprocess input columns into transformed columns."""

    # extract the columns and assign to local variables
    x = inputs['x']
    y = inputs['y']
    s = inputs['s']

    # data transformations using tft functions
    x_centered = x - tft.mean(x)
    y_normalized = tft.scale_to_0_1(y)
    s_integerized = tft.compute_and_apply_vocabulary(s)
    x_centered_times_y_normalized = (x_centered * y_normalized)

    # return the transformed data
    return {
        'x_centered': x_centered,
        'y_normalized': y_normalized,
        's_integerized': s_integerized,
        'x_centered_times_y_normalized': x_centered_times_y_normalized,
```

```
    }

# a temporary directory is needed when analyzing the data
with tft_beam.Context(temp_dir=tempfile.mkdtemp()):

    # define the pipeline using Apache Beam syntax
    transformed_dataset, transform_fn = (

        # analyze and transform the dataset using the preprocessing function
        (raw_data, raw_data_metadata) | tft_beam.AnalyzeAndTransformDataset(
            preprocessing_fn)
    )
```

Feature Selection

They talk about different ways of doing feature selection and deepen the analysis for 3 methods listed below.

Filter Methods: remove features highly correlated to each other (most used one is Spearman correlation)

Wrapper Methods:

- Forward elimination: start with one feature and add features one by one verifying the model's performance
- Backward elimination: start with all features and evaluate model performance when removing each of the features
- Recursive elimination: fit model and rank features by importance, discard least important ones and repeat

Embedded Methods:

- L1 regularization: they don't talk about it, but somehow one can eliminate features based on their score given by L1 regularization
- Feature importance: discard features scored lower by feature importance. Many times it's in-built in the ML models, such as the Random Forest classifier

Week 3 - Machine Learning Data Lifecycle in Production

This week they talked about Data Journey and Data Storage, Evolving data, and Enterprise Data Storage.

Data Journey and Data Storage

They elaborate on how Artifacts are created as the components of the ML pipeline are executed (including data and objects generated by the ML pipeline). Also, they emphasize the importance of data lineage (organization and regulatory compliance) and data versioning (version control for datasets, tools: DVC, Git-LFS).

Lastly, they provided a solution (ML Metadata from Tensorflow, MLMD) that would simplify how to track data changes.

Evolving Data

They talk about schema development and schema environments. To recap, schema contains a lot of information about the data, such as:

- Feature name
- Type: float, int, string, etc
- Required or optional
- Valency (features with multiple values)
- Domain: range, categories
- Feature default values

One application of schemas in serving environments, for example, is to check automatically if there is any anomaly found in the data (example below).

Feature name	Anomaly short description	Anomaly long description
'Cover_Type'	Out-of-range values	Unexpectedly small value: 0.

Enterprise Data Storage

They talk about Feature store, how to manage feature data, scalable and performant access to feature data in training and serving, and providing a consistent and point-in-time correct access to it.

They also talk about data warehouses and the comparison with databases (comparison below).

Comparison with databases

Data warehouse	Database
Online analytical processing (OLAP)	Online transactional processing (OLTP)
Data is refreshed from source systems	Data is available real-time
Stores historical and current data	Stores only current data
Data size can scale to >= terabytes	Data size can scale to gigabytes
Queries are complex, used for analysis	Queries are simple, used for transactions
Queries are long running jobs	Queries executed almost in real-time
Tables need not be normalized	Tables normalized for efficiency

Finally, they talk about data lakes and the comparison with data warehouses (comparison below).

Comparison with data warehouse

	Data warehouses	Data lakes
Data Structure	Processed	Raw
Purpose of data	Currently in use	Not yet determined
Users	Business professionals	Data scientists
Accessibility	More complicated and costly to make changes	Highly accessible and quick to update

Week 4 - (Optional): Advanced Labeling, Augmentation and Data Preprocessing

This week they talked about:

- Advanced Labeling
    - Semi-supervised learning
    - Active learning
    - Weak Supervision
- Data Augmentation
- Preprocessing Different Data Types
    - Time Series
    - Sensors and signals