Challenges

Data (77 times):

- (input) data quality
    - Data preparation (Deduplication very complex (tool helpful)) (4 times) (Paper 14; 33; 42; 49)
        - slicing the data (also for training) (identifying subpopulations) (3 times) (Paper 29; 31; 33)
        - Data cleaning (limited methods) (3 times) (Paper 1; 2; 33)
        - Data validation (checking if the data has the expected shape) (incomplete or incorrect datasets(2 times) (Paper 33;44)
        - Migration of data (no normalized format)(how to Representation) (Important) (Paper 9)
        - Automated data analysis (Paper 26)
    - (No list how gooddata quality looks like) (missing domain knowledge)(data sources were not “ML-ready”) (6) (Paper 3; 14; 21; 34; 36; 42)
    - Balancing (diversity to get fairness) data (also training with positive and negative samples and every minority groups) (5 times) Paper 27; 31; 32; 36; 44)
- correct use and storing of data [Legal regulations (Finding usage of specific data, specific storing(not same place like system), overview; using individual-level demographics and the need to declare why it is save and useful) (6 times); the right data for the feature (2 times)] (Paper 3; 27 twice ; 31; 34; 36; 42; 49)
- Data collection (also automated, expensive (3)) (getting (limited) training data (5)) (No Guideline (3)) (Paper 1; 2; 5; 9; 12; 26; 27; 31; 36; 42)
- Data management (keeping up to date, new kinds)(also across different industries)(sequences, categorical) (4 times) (Paper 1; 33; 36; 53)
    - Changes in data sources and distribution and schemas (detection) (compatibility) (data drifts) (6 times) (Paper 2; 34; 36; 42twice; 44;
- Data availability (“Cold-Start-Problem”) (2) (Paper 1; 54)
    - access (also conditions) to the needed data (e.g. for privacy, enterprise reasons) (Paper 3)
    - Consuming Data from others (silent update or correction) (Paper 10)
- (input) data quantity (large amount) (4 times)(Paper 3; 26; 44; 53)
- Heterogeneity in data and real world (2 times) (Paper 42; 45)
- Missing tooling for data dependencies (for code dependencies they exist) (2 times) (Paper 10 both)
- Data preprocessing (need to be automated and accelerated) (Paper 36)
- Data understanding (Paper 33)
- data model localization (Paper 2)
- Feedback:
    - “If a viewer never get serves activities of a certain type he couldn't give a feedback on such activities" (Paper 46)
    - Finding feedback loops (direct and hidden) (10 2times; 42)
    - Explicit Feedback is noisy (Paper 55)
    - Use cases that require immediate feedback (Paper 36)

Software Quality (63 times):

- Fairness:
    - A statistical definition of fairness is needed (then constraints are possible) (Paper 30)
    - Finding unfairness without help from the "outside" (Paper 31)
    - Using coarse-grained demographic information for fairness auditing (Paper 31)
    - Debugging and remediation of fairness issues once detected (Paper 31)
    - unexpected side effects of fairness interventions (Paper 31)
    - Determining the fairness of a trained model (Paper 33)
- Scalability ("streaming graph processing problem"; architecture; automated process) (4 times) (Paper 18; 46; 50; 55)
    - Generality comes at the cost of scalability (Paper 55)
- Documentation
    - poorly or missing documentation (also for decision making) (4 times) (Paper 6; 8; 40; 41)
    - Undeclared consumer (creates hidden feedback loops, hard to find) (Paper 10)
    - making clear specifications (Paper 8)
- reusability: (zusammengefasst und zusammenhang zu Dokumentation)
    - Transfering a built solution to another domain (Paper 36)
    - Useful Reusability (Paper 12)
    - Making modules reusable (Paper 22)
    - code reuse is minimal (nearly the whole System must be developed) (Paper 3)
- Metrics:
    - Missing metrics (Fairness metric; "accurately map to user satisfaction with the recommendations") (3 times) (Paper 31; 43; 55)
    - Optimizing a system to many metrics at the same time (Paper 55) (a solution)
    - principles to assure the safety and stability is missing (Paper 8)
    - Defining single number evaluation metric for ML projects (Paper 27)
    - Measures to quantify the predictive power including accuracy, precision and recall (Paper 27)
- Maintainability:
    - Multiple language smell (leads to difficulties in the integration test) (2 times) (Paper 34; 37)
    - Configuration dept (e.g. tuning hyperparameters; modifying correctly) (2 times) (Paper 12; 37)
    - Lack of declarative abstraction for the whole ML pipeline (glue code) (Paper 34; 37)
- Evaluation:
    - Evaluating the accuracy of the data set (Paper 8)
    - Evaluation of "the benefits only by using it in actual business" (Paper 8)
- Analysing:
    - performing sliced analysis (Paper 2)
    - Simulations or virtual reality as basis for certification (Paper 26)
- Reproducibility (Recreating a request on log data basis) (Paper 28)
- Outputs must be robust and reliable (Paper 3)
- fully traceable, auditable procedures for software development is needed (Paper 3)

- Quantification of quality targets (Paper 27)
- regulations that defines how safe is safe enough (Paper 26)
- Technical transparency (Paper 36)
- Robustness of the system (Paper 36)
- Creating system that meets all requirements to run stable and reliably to production scenarios but is still flexible enough to allow rapid experimentation and algorithm development (Paper 45)
- Stability of the system (Paper 36)
- Validation of deployed ML solutions (Paper 36)
- Explainability (Paper 39)
- security and confidentiality guarantees through the infrastructure (Paper 39)
- Coincidental correctness (Paper 40)
- (in-)Consistency (in a service itself; top label's confidence) (2 times) (Paper 41 both)
- "Providing transparency with regard to model" (Paper 36)
- Program comprehension (Paper 27)
- Explain single predictions of the model (Paper 27)
- Correctness with observable effects cannot be deterministically defined (TensorFlow and a ML characteristic) (Paper 40)
- Preventing structural inefficiency (Paper 49)
- Input signals (data) are unstable (qualitatively or quantitatively changing over time)(2) (Paper 3; 10)
- Express behavior effectively (Paper 10)
- Interpretability (Paper 1)
- Incompatibility (Conflict with the OS, hardware conflicts, "Conflicts with algorithm model version", platform software problems) (Paper 6)

Testing (20 times):

- Experiments:
    - designing experiments (Paper 2)
    - repeat experiments (Reproducibility; no standard)(hard to reproduce without versioning and something like a protocol) (Paper 3; 9)
    - Trekking the experiments (Paper 9)
    - Find dead experimental code paths (Paper 37)
    - Rapid experimentation frameworks (Paper 55)
- Testability (Missing methods/tests for user-friendliness and robustness of functions) (problem with probabilistic correctness) (5 times) (Paper 7; 9; 26; 39; 41)
- testing only some modules (Paper 3)
- testing "using traditional software engineering methods" (Paper 3)
- verify the results of testing (Paper 3)
- Coverage of tested code (Paper 8)
- Real-world testing (Paper 26)
- Automated testing on proving ground and integrated into fully automated vehicles (Paper 26)
- Testing is expensive (Paper 42)
- Finding the reason for changing in backtesting results (Paper 34)
- Other evaluation method then trial and error because of losing too much time (Paper 9)
- Simulation:

        - Simulation environment (accurate sensor/vehicle models, the relying real-world data from real traffic) (Paper 26)

<u>Infrastructure (22 times):</u>

- Tooling:
    - Missing tool to support the collection of balanced or representative datasets (Paper 31)
    - Missing tools (static analysis of data dependencies(10); support fairness and equity in downstream (31); visualization (and their techniques) (Paper 1;33; 36); simulation tools testing (Paper 9); less than “normal systems” (26)) (7 times)
- Logging and monitoring
    - Logging at all (also for many entities and the best way) (2 times) (Paper 26;35)
    - comprehensive live (also in production) monitoring (also models) (2 times) (Paper 2; 10)
    - complex and poor logging mechanisms/techniques (Paper 2)
    - Large amount of logging data (massive and distributed; large traffic) (Paper28;52)
    - Sharing and tracking techniques (are missing) (Paper 42)
- Pipeline (6):
    - Implementing Pipeline (end-to-end or automated) (Paper 1; 2; 36; 45)
    - Maintaining, managing and improving the required ML pipelines (different backgrounds of people who have to handle this) (Paper 33; 34)
- long deployment cycle (Paper 5)
- Building the Infrastructure (e.g. experimentation infr.; combined testing in simulation and real environments) (3 times) (Paper 2 two times; 26)
- Working with the already available infrastructure on the customer side (Paper 36)

<u>Model (17 times):</u>

- Model Debugging and Interpretability (e.g. Understanding the results of the model) (2 times) (Paper 1; 9)
- Definition (and explaining) of ML models (e.g. what has been learned) (Paper 27; 34)
- Choice of Model (Which will perform best for given problem) (2 times) (Paper 49; 55)
- Correct model parameter or structure (2 times) (Paper 40; 49)
- Model Evolution, Evaluation, and Deployment (Paper 1)
- Pre-trained models that work well in consumer context may not be as effective in enterprise context (Paper 39) (no solution)
- Invalidation of models (Paper 2)
- Scaling the models (Paper 2)
- Backwards compatibility of trained models (Paper 34)
- Understanding of nuances of the model (missing knowledge) (Paper 9)
- Management of models (Paper 34)
- Migrating the model to the final software solution in production (Paper 38)
- Bilateral communication for the models using same upstream data source (Paper 30)

<u>Debugging (17 times):</u>

- Behaviour of a failing system is not always the same (error will not always raise in the same iteration (because of the stochastic nature, so hard to find bug); no fault could be seen; finding the failing part) (5 times) (Paper 1; 12; 30; 40 2-times)
- Finding the (exact) reason for a (specific) bug (5 times) (Paper 8; 11; 12; 13; 49)
- Changing anything changes everything (3 times) (Paper 10; 29; 32)
- Versioning (version control) of ML components and data (2 times) (Paper 9; 41) (solution)
- small improvements can have meaningful impact at scale and need large amounts of data to be observed with high confidence (Paper 30)

Training (8 times):

- Overfitting (4 times) (Paper 34; 35; 43; 48)
- time consuming (training a full CNN to learn a good representation) (Paper 53)
- large amount of data required (Paper 53)
- Bottleneck by reading all features as part of each training example from the disk (Paper 35)
- Training serving screw (Paper 21)

Optimization (7 times):

- Performance optimization (also mobile) (Paper 6; 24)
- Finding features that are not effecting the results (Paper 37)
- complexity of optimization techniques (Paper 1)
- Performance characterization and analysis (fragmentation of mobile ecosystem) (Paper 24)
- finding minimal architectures without reducing model accuracy (Paper 33)
- Personalization (Paper 46)

Knowledge (6 times):

- Education and Training of the team (Paper 1)
- "Gap between the engineering team and the customers" (Insufficient Understanding of Customers (e.g. 100% accuracy; too much expectations regarding the functionality; easy to start) (4 times) (Paper 8 everything)
- an average developer and product manager in such enterprises couldn't be familiar with the machine learning technology and it's nuances (Paper 39)

Standardization (5 times):

- No standardized way to store and manage resulting experimentation data and artifacts (Paper 34)
    - "lack of standardized approaches for reproducing model selection experiments quickly" (Paper 2)
- "standardized architectures for local solutions" (Paper 36)
- Cloud solutions offer standardized solutions for stability (Paper 36)
- "standardization and user-friendliness for application of ML models" (Paper 36)

API (Also ML algorithm and frameworks) (5 times):

- Choose the right ML algorithm/features (e.g. for developing a model) (Paper 9)
- No possibility to 'plug-and-play' for intelligent services (Paper 41)

- Changes in the API/Framework (e.g. features disappear, changes in results) (Paper 33)
- solid algorithmic techniques (Paper 55)
- Handling impedance mismatches that arise from crossing boundaries between different systems and frameworks (Paper 28)

Development (4 times):

- "managing design trade-offs in customization of platform functionalities" (Paper 2)
- ad-hoc patterns used to develop ML systems (Paper 4)
- Implementation activities are ad-hoc (Paper 8)
- Programmability by using GPUs and DSPs (Paper 24)

General/others:

- Get the ground truth (against which to test) in a perfect way (find a way to get it) (2 times) (Paper 9)
    - Formulating the Problem (Paper 2)
- Mobile: (as challenge at all)
    - deep learning for mobile platforms (strongly limited resources)
    - Performance optimization for mobile (2 times) (mobile inference performance exhibits significant variability, even across the same device running the same software stack) (Paper 24 everything)
- Partitioning a graph/system over several machines (Paper 43)
- Coupling between stages like feature engineering, model definition and model training and evaluation (Paper 37)
- prediction bias (Paper 37)
- Incurred bias because of domain differences (Paper 48)
- The as a service movement (Paper 39)
- Structure inefficiency (Paper 40)
- running large scale data processing workloads on compute clusters of varying sizes with the same code (Paper 45)
- “Heterogenous customer base" (Paper 48)
- "rapid ingestion of edges as well as high-volume reads" at the same time (Paper 50)
- Uncertainty in Input-Output Relationships (Paper 8)
- "enforce strict abstraction boundaries" by prescribing intended behavior (Paper 10)
- Large computational demands for delivering inference (Paper 24)
- Performing embedded devices as all processing on-device with high performance delivery (Paper 24)
- Threading dataflows across multiple interfaces (Paper 28)
- "Automatically generating fixes and playbooks" (Paper 33)
- "Determining the best embedding that should be used on a feature by searching over an available set of pre-trained embeddings" (Paper 33)
- Creating feature extraction code (Paper 34)
- "Varied Perceptions" (Paper 1)