Data: 48

- Carefully slicing the full data into training, test and validation set (4 times) build of each slice "datasets of different sizes from small-scale to large-scale" (2 times) (Paper 12; 18;21; 29; 34; 52)
    - Divide "Evaluation data in two datasets: Satisfactory and Unsatisfactory" (Paper 11)
    - Do not use content of the training or validation data for testing, but a subset of the whole data (Paper 15; 18; 21; 34) (to fight the overfitting)
- Getting Data:
    - "information about previous such actions contain highly relevant information for predicting future actions" (Paper 55)
    - Track at which position an interesting object must be placed that the user would click it (Paper 55)
    - Record every action from the customer (Paper 55)
    - "assign every user who visits an advertiser's website a positive label" and assign every active user during this time period an negative label (Paper 48)
- Preprocessing before using the data (training data not contain "protected" characteristics) (feature engineering, data validation, data shuffling) (3 times) (Paper 27; 38; 49)
    - Downsampling the dominant classes (3 times) (Paper 17; 18; 54)
    - Deterministic shuffling (randomizing) of the input data (reducing serving bias) (Paper 17; 46)
    - Control evaluation and training samples carefully (Paper 34)
- Document what data is available and how it is organized (metadata) (Paper 28; 55)
- keeping only the last few days in high granularity the older one can have a lower granularity. So it is possible to fall back to finer granularity for the last few days (Paper 44) (solution)
- Large amount of data that is not the optimal is better than small amount of data from the optimal (Paper 48) (solution)
- Reduction of the scope (e.g. documents with activity within last 60 days, used features) (Paper 21) (solution for large amount of data)
- Receive "data from third party vendors or from machine learning competition" (Paper 9)
- enforcing simple, generic schemas for all needed data to track (Paper 28)
- If much data is available it is possible to reject data flatly if it has an incompatible schema. (Paper 44)
- Improve data quality with Domain knowledge and take freshness and diversity into account (Paper 36)
- Training sets should contain positive and negative labeled data (Paper 54)
- Set of Level 0 and Level 1 Data Flow Diagrams to identify what the data could be used for (Paper 23)
- Map manually "the source data into a stable, well-thought-out schema and perform whatever transformations are necessary" (Paper 44)
- key questions for data preparation ("what features can be generated from data, what are the properties of the feature value, what are the best practices to transcode values) (Paper 33)
- "calculating statistics from training data, and then adjusting them as appropriate based on domain knowledge." Write down expectations and compare them to the data (Paper 51)
- Data modeling with multiple iterative training rounds (Paper 12)
- To improve fairness, the data is an important place (Paper 31)
- Specify requirements and data quantity on the diversity of the examples (Paper 27)
- Create mappings to overcome the problem of missing values (Paper 54)
- Transfer the learning from long views to avoid overfitting (Paper 35)
- Using traffic light (not so much Data Traffic) systems for visualization (Paper 36)
- Feedback (to and from the user):
    - If a system is not confident it should display this or give an error message like "too many objects" (Paper 42)

- If any input data is below the information of training this should be displayed and not try to solve it and fail (Paper 42)
- create a method for explicit user feedback (Paper 23)
- Include explicit feedback (Paper 55)
- Using implicit and explicit feedback (Paper 52)
- Using precision alerts (could be helpful to fix problems), but first the anomalies that worse the accuracy in production (Paper 33)
- Triggering configurable alert if a job fails (only the failed subgraph needs to be restarted) (Paper 44)

Testing: 44

- **testing methods:**
  - A/B tests/bandit test (also parallel)(online on live traffic)(for tuning) (9 times) (Paper 23; 32; 34; 43; 47; 51; 52; 54; 55)
  - Sanity checks (5 times) (Paper 21; 33; 34; 35;36)
  - Use basic integration tests (also for whole ML pipeline) (2 times) (Paper 44;51)
  - Fault injection test (Paper 8)
  - Online performance test (Paper 35)
  - K-fold cross-validation for testing (Paper 12)
  - Desktop bucket test (Paper 46)
  - consistency checks for the input data (also automated) (Paper 36)
- **Use Experiments:**
  - Controlled user experiments (Paper 55)
  - Use 5 steps for developing an experiment (Start with hypothesis, Design a test, Implement the test, Execute the test, analyze the test) (Paper 55)
  - offline experiments on retrospective data (Paper 43)
  - Small scale A/B experiments/test using an intentionally degraded model (Paper 51)
- **Evaluation:**
  - Leave- one-feature-out evaluation (to remove unnecessary features) (Paper 10) (challenge solution)
  - "To conduct evaluation for visual search, we used the image annotations associated with the images as a proxy for relevancy. This approach is commonly used for offline evaluation of visual search systems [...] in addition to human evaluation." (Paper 53)
  - Using the same code for evaluating metrics (throughout time and across different models) (Paper 34)
  - Using qualitative and/or quantitative KPI evaluation (Paper 23)
  - Using replay methodology for unbiased offline evaluation of online serving schemes to evaluate the model (Paper 46)
  - Evaluate course-to-skill mapping with skill coverage and precision and recall (Paper 54)
  - "human-driven evaluation for more sensitive data categories" (Paper 1)
- Test if all features are beneficial ("by computing correlation coefficients, by training models with one or two features, or by training a set of models that each have one of k features individually removed") (Paper 51) (solution)
- After offline testing validated a hypothesis it should be tested online (e.g. online A/B tests) (Paper 55)
- Test the full system (vehicle) (Paper 26)
- Each time one model is re-trained, it must pass two different quality checks (paired t-test, rank of correlation between sets of model coefficients in successive buildings (Paper 20)
- Test multiple models on live traffic (Paper 21)
- Resource usage test (finding which resource could be a problem) (Paper 8)

Quality: 28

- Reusability (Challenge solution)
  - Reusing the structure and the data-sets of a neural network (Paper 22)
  - Reuse as much code as possible (Paper 23)
  - Identifying shared resources for multiple KL products (reusability) (Paper 23)
  - Using “SimilarLooks” feature to solve a challenge (Paper 53)
  - Use solution for slightly different problem and learn a small correction to create fast solution (if possible) (Paper 10)
  - make labeled data available to everybody and reuse it as much as possible (Paper 1)
- Reliability (Check the model by missing a specific source)
  - If data is currently not available, a straightforward code change change it and the data flows automatically and reliably to all offline systems (Paper 44)
  - Make easy and quick rollback to older versions available (2 times) (Paper 36; 51)
- Reproducibility (Use a snapshot of the whole System to reproduce results) (Paper 12) (soluti)
  - Experiments reproducibili<ty through documenting exact version (Hardware, Platform, Source Code,Configuration, Training data, Model state (Paper 12)
  - Record every decision and every changed decision (Paper 23)
  - Storing all the components defining a model (Paper 34)
- Specification and documentation :
  - "Every model specification undergoes a code review and is checked in to a repository:" (Paper 51)
  - Make a neat documentation to all models including older versions (Paper 36)
  - Component fixing tasks should be well formulated (Paper 11)
  - Requirements:
    - Move Requirements to the training and architecture design phase (Paper 8)
    - Quantitative measures comprise requirements (Paper 12)
- Versioning (possible Challenge solution):
  - use common versioning repositories (Paper 1)
  - "each dataset is tagged with information about where it originated from and which version of the code was used to extract it" (data versioning) (2 times) (Paper 1 both)
- Fairness:
  - Using professionals using their "on the ground" knowledge to improve fairness (Paper 31) (solution)
  - change system design to address fairness issues (Paper 31)
- Validation:
  - re validating the model after data changes, methods improvements or software dependency changes (Paper 34)
- Risk management ("list risks, identify their severity, and have contingency plans ready) (Paper 23)
- Compatibility target system for the "lowest common denominator" (e.g. the mobile CPUs) (Paper 24)
- Analysis:
  - Feature-based Analysis (Paper 33)
  - Data Lifecycle Analysis (Paper 33)

Metrics/Measures:25

- Offline metrics improvement, not necessary means online metrics improvements (2 times)
- (Offline) normalized discounted cumulative gain (NDCG) measurements as ranking measure (2 times) (Paper 52; 55)
- Compute a host of metrics at the same time to avoid multiple scans over the data, and persist the resulting evaluation data (Paper 45)

- Close-up rate (CUR) and click-through rate (CTR)(2 times) (helpful for search ranking, recommendations systems and ads targeting) (Paper 52; 53)
- Use metrics like conversion rate (Paper 52)
- Most-Recently-Used (MRU) heuristic (For all classes of users) (to find a trend for a user) (Paper 21)
- Measuring the engagement using the hit-rate (Paper 21)
- "classification metrics such as accuracy, precision, and recall" (Paper 55)
- "regression metrics such as RMSE" (Paper 55)
- Training time and GPU utilization ratio as metric to find performance bug (Paper 13)
- "Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), or Fraction of Concordant Pairs (FCP)" as possible metrics (Paper 55)
- follow-through-rate (Paper 43)
- (Offline) normalized discounted cumulative gain (NDCG) measurements as ranking measure (2 times) (Paper 52; 55)
- measuring the send-to-open rate (Paper 54)
- two-fold cross-validation for performance measure (Paper 48)
- Progressive validation (also online loss) (Paper 29)
- Cost measure (“consider not only added inference latency and RAM usage, but also more upstream data dependencies, and additional expected instability incurred by relying on that feature.") (inference latency and RAM usage, upstream data dependencies, additional expected instability incurred by relying on that feature) (Paper 51)
- Continuous quality measures (before and after fix) (Paper 11)
- Evaluate fixes by humans with the metrics: "accuracy (1-5 Likert scale), detail (1-5 Likert scale), language (1-5 Likert scale), commonsense (0-1), and general evaluation." (Paper 11)
- Measure the coverage (Paper 21)

Infrastructure:25

- Monitoring:
    - Monitoring data stream ("rate of activity on each of our action pixels" and "quality of mapping pixels" (external data sources) (Paper 20)
    - Monitor Interfaces (Paper 20)
    - Dedicated and continuous monitoring (Paper 36)
    - Monitoring the quality, usefulness and provenance of records (Paper 28)
    - Monitoring of the models (Paper 34)
    - Monitoring and testing (monitoring: Prediction Bias, Action Limits, Up-Stream Producers) (Paper 10)
- Logging: (solution)
    - Using protocol buffer and thrift messages to log events (Paper 18)
        - Store logs as serialized Thrift instances (Paper 35) (solution)
    - Logging samples of serving traffic (Paper 51)
    - Logging high-level actions ("create, open, edit, delete, rename, comment, and upload events") (Paper 21)
- If possible use a single server (easy to scale, just copy) (Paper 43) (solution and important)
- Persistence layer (for fast linear reads and writes), Network layer (for zero-copy data transfer) (Paper 44)
- Using a server to retrieval and scoring the models (Paper 35)
- Streaming examples from disk or over the network (Paper 29)
- Two kafka cluster (one for the online service and a copy for offline prototyping and data loading into hadoop) (Paper 44)
- Processing logs using a Spark pipeline to create training data (Paper 35)
- loosely coupled components (running independently from each other) (Paper 45)

- Create the ML service as a separate cloud service (deployed and operated along the BPMS) (Paper 39)
- Running several analytics pipelines at the same time (Paper 38)
- Deliver recommendations to the front-end via an online REST Endpoint (Paper 54)
- Running the full offline job pipeline at short, regular intervals (Paper 52)
- two neural networks (one for candidate generation and one for ranking) (Paper 47)

- Data extraction pipeline (discards data for quality or policy reasons) (Paper 21)
- automate the training and deployment pipeline (Paper 1)
- Building a supervised platform (Paper 54)

Optimization:20

- Tuning Hyperparameter using automated meta-optimization methods (replacing) (3 times) (Paper 15; 40; 51)
- Encoding values with fewer bits (32 or 64-bit floating values) (Paper 29) (Solution memory)
- Express relative changes as a percent range (Paper 29)
- "Comparing the relative changes of accuracy and loss between iterations" (Paper 40)
- Before deploying for edge inference, optimization techniques (quantization) can be applied to Optimizer (Paper 24)
- Optimize trained models with " aggressive quantization and weight/channel pruning" (Paper 25)
- Comparing performance of ML models only when it has been trained and evaluated using the same (Paper 34)
- Manipulating training data instead of ML algorithm (also for fairness) (Paper 27)
- Normalize (continuous) features (Paper 35; 47)
- Easier to improve a system that provides adequate performance at scale (Paper 20)
- "Similarity is also an important aspect of personalization" (Paper 55)
- Iterative optimization (Paper 12)
- "freeze or write out the weights after just one iteration" (Paper 12)
- Use the multithreading to perform many random walks in parallel (Paper 43)
- Solve the inner optimization problem as quickly as possible (solving to a fairly low tolerance and apply a hard limit for the number of iteration) (Paper 45)
- "Comparing overall accuracy and loss with fixed thresholds" (Paper 40)
- Using GPU-based inference to reduces latencies (Paper 53)

Model: 16

- Training many similar models (to get the best model and save time) (Paper 29)
- Trigger retraining or re-modelling if new events occur that the model have not been trained for (Paper 34)
- Build initial models as simple as possible by resulting reasonable results while collecting more data (Paper 36) (cold-start problem solution)
- "Combining many independent models developed by different teams that joined forces" (get the best results) (Paper 55)
- First filtering with simple features/models, then filter the rest with more complex features/models (Paper 19)
- constantly updating the model by selecting and relabeling a completely random subset of incoming content (Paper 29)
- Isolate models and serve ensembles as mitigation strategy (Paper 10)
- Three approaches to improve model accuracy (increasing training data, refining feature sets, and changing model architectures) (Paper 25)
- High-level linear ensemble model ("features computed from the user's cookie, form the basis") (Paper 20)

- Single value structure (store only one coefficient value for each coordinate shared by all models)
- Selling pre-trained models (get money for the own projects)
- "Support an extension to local learning which allows models to share statistical strength between items" (Paper 45)
- "Model quality is sufficient on all important data slices" (with release tests) (Paper 51)
- "The model has been tested for considerations of inclusion" (Paper 51)
- models including typical Logistic Regression, Support Vector Machines, or Gradient Boosted Decision Trees (Paper 55)
- Use existing systems for tracking and storing models (feature transformation with model parameters is also possible) (Paper 34)

Tooling and Algorithm: 15

- Using Stochastic Gradient Descent (Paper 21)
    - to minimize the loss function (Paper 19)
    - to optimize the parameter weights (Paper 16)
- Visualization tools such as Facets (Paper 51)
- Branch coverage with tooling support (Paper 7)
- Static code analysis with tooling support (Paper 7)
- "whole infrastructure runs across the public Amazon Web Services cloud" (Paper 55)
- Use of a cascade procedure to rank features. If all stages accept a feature it will be accepted otherwise it will be dropped (Paper 19)
- For logistic regression use the Alternating Direction Method of Multipliers (ADMM) (Paper 46)
- AdaGrad as a robust learning algorithm (for deep networks) (Paper 21)

- "naive Bayes and logistic regression trained with stochastic gradient descent" (Paper 48)
- Use " a stacked ensemble to estimate a low-dimensional function" (Paper 48)
- Follow the regularized leader algorithm (Paper 29)
- Bloom Filter inclusion (Paper 29)
- Matrix Factorization and Restricted Boltzmann Machines (Paper 55)

Training:13

- Training is done offline (also in the cloud)(label generation, Model optimization, Feature Generation Pipeline) (2 times) (Paper 24; 55)
    - Perform additional training online (Paper 16)
- Shuffling the data just prior to training (Paper 18) (also other shuffling is mentioned)
- reproducible training ("ensembling models can help") (Paper 51)
- Model training with TensorFlow (Paper 35)
- Training on single machine by "increasing model replicas and employing data parallelism across GPUs" (Paper 16)
- Distributed training (Paper 16)
- Add random negative training data to get balanced training sets (Paper 54)
- To train, sampling negative classes from the background distribution and then correct this sampling via importance weighting (Paper 47)
- Use the direct feedback to train the model (Paper 47)
- Using roll-ups or drill downs to find problematic slices in training data (Paper 33)

- Making model training more robust to skew (Paper 33)

Frameworks:13

- for effective serialization of the data Avro is a possible framework (challenge solution) and also for the separation of the logical and physical data representations (Paper 28; 44)

- Using Caffe2 features to exploit "performance optimization opportunities" (for production) (Paper 16; 24)
- Using Kafka for infrastructure of the data (recognizing shifts within input data) (Paper 36; 44)
- Using Pig for workflow (a high level dataflow language) (also for preparing and sampling test data) (Paper 18; 44)
- Using GraphJet: (as good practice for graphs)
    - to manage "a dynamic, sparse, unidirected bipartite graph" (Paper 50)
    - "temporally partitions the adjacency lists by periodically creating a new index from scratch" (Paper 50)
- For better Performance and fitting try to extend frameworks (“building on top” of them) (Paper 12)
- Apache solr for indexing data (Paper 52)
- With a MapReduce job many models could be constructed and all could be trained independently on each partition (Paper 18)

Development: 10

- Using Scrum or prototyping (2) developing (Paper 5; 23; 44)
- Ensure the vocabulary used by a particular service is right before implementation (Paper 41)
- Using the different techniques for ML component and “normal” component development (Paper 2)
- Evaluate the system after each fix (Paper 11)
- Using "a two-stage transfer learning approach" ("learning multiple candidate mappings and then weight and combine them") (models from the first stage could be used directly in production) (Paper 48)

Debugging:10

- fix language by using the likert scale (also the language fluency) (Paper 11)
- "Language fixes are generally more effective than the commonsense fixes" (Paper 11)
- A fix workflow (1. Current system evaluation; 2. Component fix simulation; 3. Fix workflow execution; 4. After-fix system evaluation) (Paper 11)
- Dynamic computation as debugging (on sample data to verify the idea) (Paper 12)
- "print parameters and gradients to console" to debug (Paper 13)
- The model allows debugging by observing the step-by-step computation of training or inference on a single example (With TensorFlow debugger) (Paper 51)
- change if conditions (Paper 6)
- "Object fixes are more effective than the activity ones" (Paper 11)
- Introducing human in the loop for simulating component fixes and "evaluating the systems before and after the fixes" (Paper 11)
- "Entanglement detection requires the execution of workflows with different combinations of fixes to measure the individual and the joint effect of component fixes." (Paper 11)

Design and Architecture:9

- High level architectural design is relatively fixed (data collection, data cleaning, feature engineering, data modelling, execution and deployment) (Paper 12)
- layered Architecture:
    - Generic three layer architecture (Paper 55)
    - two layer architecture (online and offline) (Paper 54)
- Using layer architecture (online, offline) (Paper 44; 54; 55) www
- solr with slave/master architecture (slave mirror the content available from master) (Paper 52)
- "[M]ake design decisions that benefit the majority of the models without severely damaging any of them" (Paper 48)

- Modularity as fundamental platform design principle (Paper 54)
- Using backdoor-based architecture (to "frees up the recommendation engine from the complexity of managing extrinsic logic") (Paper 52)
- "Vertical-designed approach to offload DNN models using the BoltNN DSP backend for its VR platforms" (Paper 25)
- Simplify error analysis and debugging by modularization in a conventional, layered, and tiered software architecture (Paper 1)

Memory:5

- To not create oversized adjacency lists, start with a small size. Each time the system runs out of space, create a new double sized list. (Paper 50)
- After all records passed the learner, a storage function writes the learned model to disk (Paper 18)
- Use one writer, but multiple reader instances (Data Access) (Paper 50) (solution)
- Each predictor and trainer instance is constrained to a fixed maximum memory usage and the same for the model size (To solve the problem of a strongly limited memory) (Paper 39)
- "only the latest index segment is actively being written into [...] the remaining are read-only" (Paper 50)

General:26

- Three possible computations (offline, online and nearline) (nearline is the best way for AI) (Paper 55)
    - computing as much as possible offline (e.g. pre-filtering of data and models or evaluation or changing of production parameters) (Paper 55)
    - offline computation as fallback (if online computation is not possible) (Paper 55)
- Graph:
    - Graph database (Paper 43)
    - Shuffle the graph structure at each iteration (Paper 43)
    - Cassovary in-memory graph processing engine (Access via vertex-base queries) (Paper 50)
    - Storing Real-Graph on HDFS ("freedom to incorporate as many signals as [...] available) (Paper 50)
    - Put the entire graph in memory on a single machine (if possible) (Paper 50)
- ad:
    - "the next best outcome is usually a visit to the brand's website following an ad impression" (Paper 48)
    - First rank the data, then permute the first k results uniformly at random (Paper 46)
- Knowledge:
    - Train the employees by host international conferences, hosting weekly forums (Paper 1) (challenge)
- Formulate the problem after analysing the data (Paper 28)
- run queries in a distributed fashion (Paper 55)
- "Treating ranking as a classification or regression problem is known as a pointwise approach in the family of machine learning techniques known as Learning to Rank." (Paper 55)
- Learning from retrospective meetings (e.g. talk about problems) (Paper 23)
- Different operation modes for the system (e.g. normal, debugging) (Paper 45)
- Ensembles lead to higher accuracy (learning in parallel and no single machine onto which all data must be shuffled) (Paper 18)
- Fixed thresholds in dynamic systems (Paper 10)
- Showing the user the most broadly popular of a new genre/area before the smaller niches and show random values this to a small fraction of users (to not reduce the hit rate)(Paper 47)
- Using Thumbnails to visualize documents in Quick Access (optimization) (Paper 21)

- "taking a logarithm of features with very large values" (Paper 21)
- Human Evaluation (Using score cards) (Paper 1)
- "The use of both text and visual signals for object detection and localization is widely used for Web image retrieval and categorization" (Paper 53)
- A regularization parameter personalizes large coefficients to reduce overfitting (Paper 48) (Solution)

- "Keep consistency between GPU and CPU" (Paper 6)