Production: Model Development

```
$ echo "Data Science Institute"
```

Agenda: 5.1 Model Development and Offline Evaluation

- Model Development and Training
- Ensembles
- Experiment Tracking and Versioning
- Distributed Training
- AutoML
- Model Offline Evaluation

Agenda: 5.2 Experiment Tracking

- Observability and telemetry
- Docker and Portability
- Experiment Tracking in Python
- Experiment Components

Slides, Notebooks, and Code

- These notes are based on Chapter 6 of *Designing Machine Learning Systems*, by Chip Huyen.

Notebooks

- `./notebooks/production_5_model_development.ipynb`

Code

- `./05-src/credit_experiment_*.py`

Our Reference Architecture

The Flock Reference Architecture

Model Development and Training

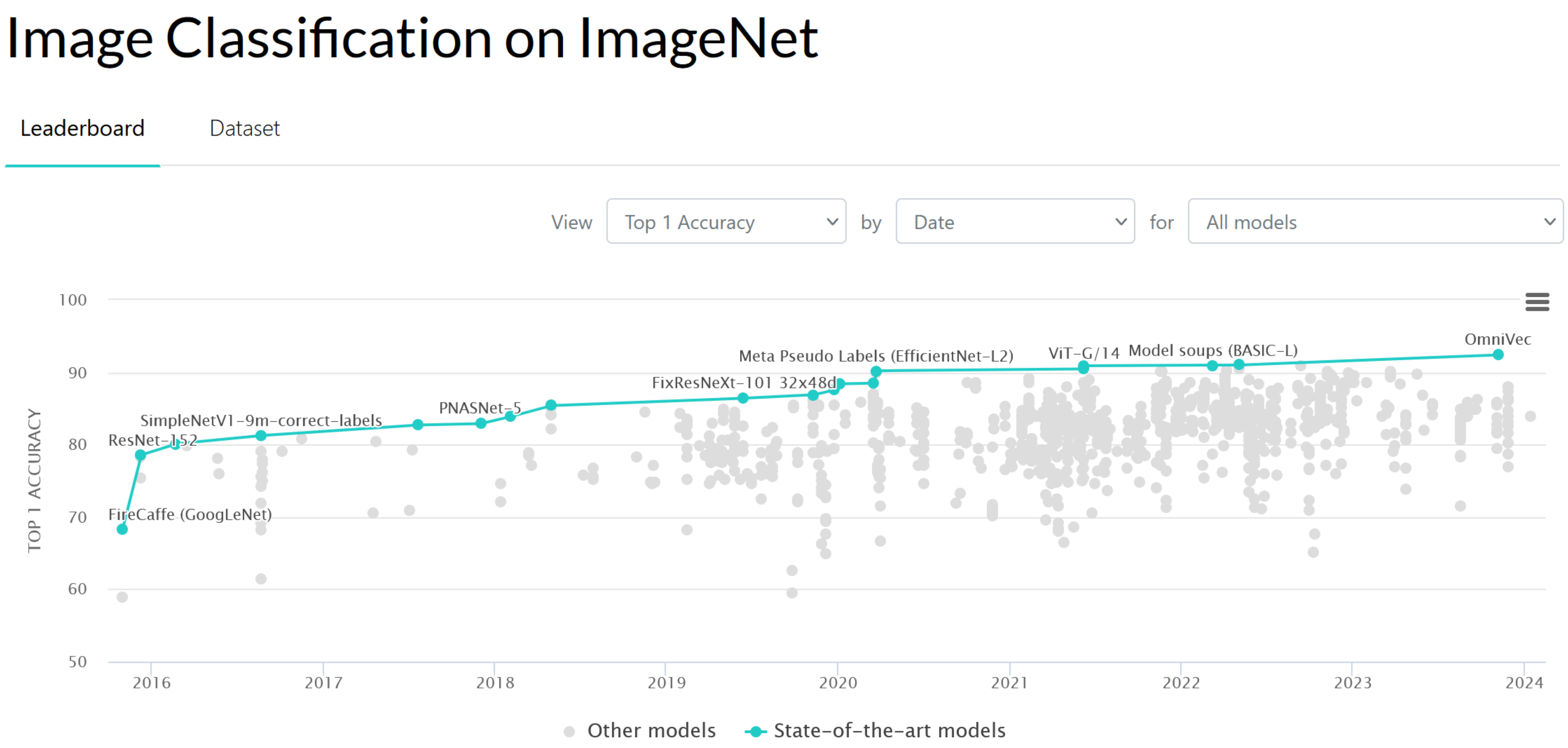
Evaluating ML Models

- Evaluating ML models in production is a multidimensional problem.
- Model performance (of course) is important, but so are how long it takes to train, latency at inference, (cost of) compute requirements, and explainability.
- Different types of algorithms require different numbers of labels as well as different amounts of computing power.
- Some take longer to train than others, whereas some take longer to make predictions.

Guidance for Model Selection (1/3)

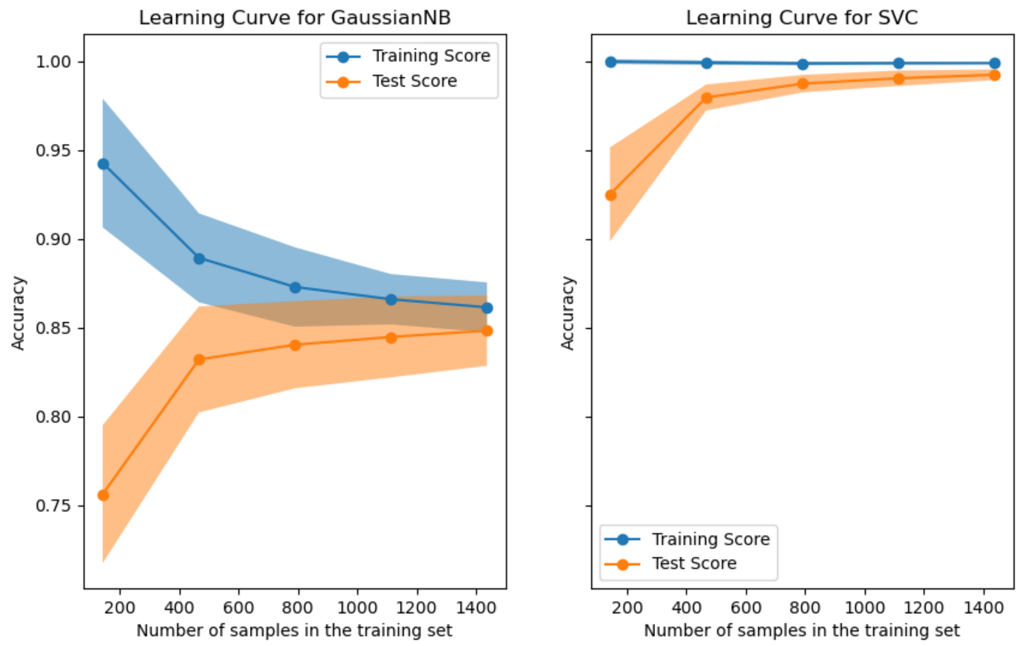
- Avoid the state-of-the-art trap
    - Researchers evaluate models in academic settings: if a model is state-of-the-art, it performs better than existing models on some static dataset.
    - It is essential to remain up to date but solve the problem first.
- Start with the simplest models
    - Simple is better than complex: easier to deploy, easier to understand, and serve as a baseline.
    - Easier to deploy: speeds up the experimentation cycle.
    - Easier to understand: adds complexity as needed.
    - Baseline: simple models serve as a starting comparison point for model development.

Guidance for Model Selection (1/3)



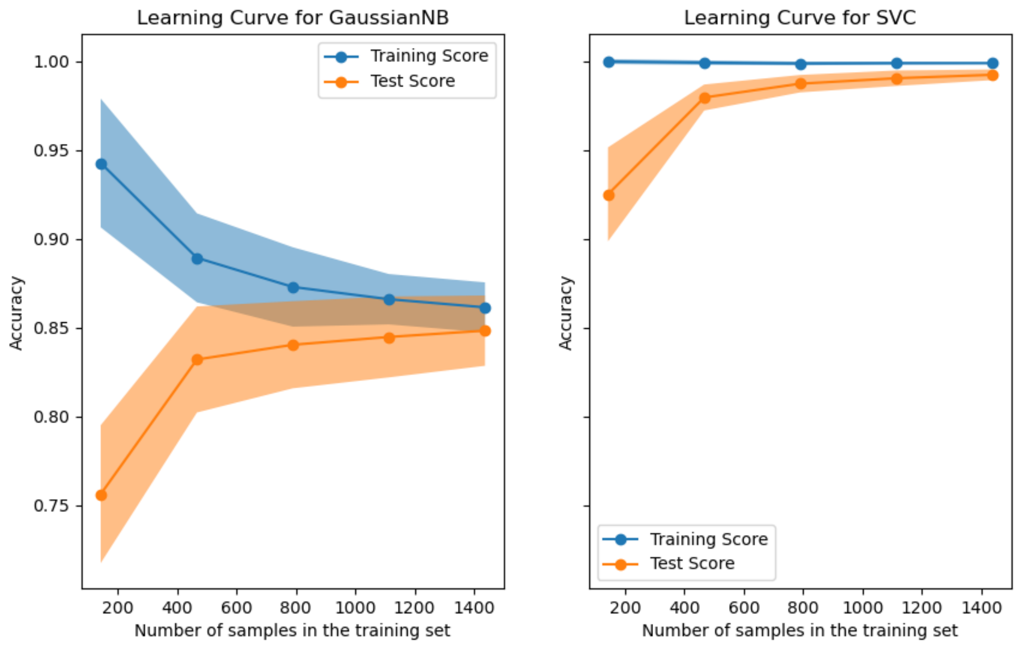
Guidance for Model Selection (2/3)

- Avoid human biases in selecting models
  - Human biases can be introduced throughout the model development process.
  - Experiment methodically and store results.
  - Any model has three components: algorithmic logic, code, and data.



Guidance for Model Selection (2/3)

- Evaluate good performance now versus good performance later
  - Using learning curves is a simple way to estimate how your model's performance might change with more data.
  - While evaluating models, consider their potential for improvement and how easy/difficult it is to achieve.



Guidance for Model Selection (3/3)

- Evaluate trade-offs
  - False positives vs false negatives: reducing false positives may increase false negatives and vice versa.
  - Compute requirement and model performance: a more complex model may deliver better performance, but at what cost?

Guidance for Model Selection (3/3)

- Understand your model's assumptions
    - Every model comes with its assumptions.
    - Prediction assumption: every model that aims to predict an output Y from an input X assumes that it is possible to predict Y based on X.
    - Independent and Identically Distributed: neural nets assume that examples are independent and identically distributed.
    - Smoothness: supervised learning models assume that a set of functions can transform inputs into outputs such that similar inputs are transformed into similar outputs. If an input X produces Y, then an input close to X would produce an output proportionally close to Y.
    - Linear boundaries, conditional independence, normally distributed, and so on.

Ensembles

The Wisdom of the Crowds

> “Aggregating the judgment of many consistently beats the accuracy of the average member of the group, and is often as startlingly accurate […] In fact, in any group there are likely to be individuals who beat the group. But those bull’s-eye guesses typically say more about the power of luck […] than about the skill of the guesser. That becomes clear when the exercise is repeated many times.”

> (Tetlock and Gardner, 2015)

Ensembles

- Ensemble methods are less favoured in production because ensembles are more complex to deploy and harder to maintain.
- Common in tasks where small performance boosts can lead to substantial financial gains, such as predicting the click-through rate for ads.
- Ensembles perform better when underlying classifiers are uncorrelated.

Leaderboard

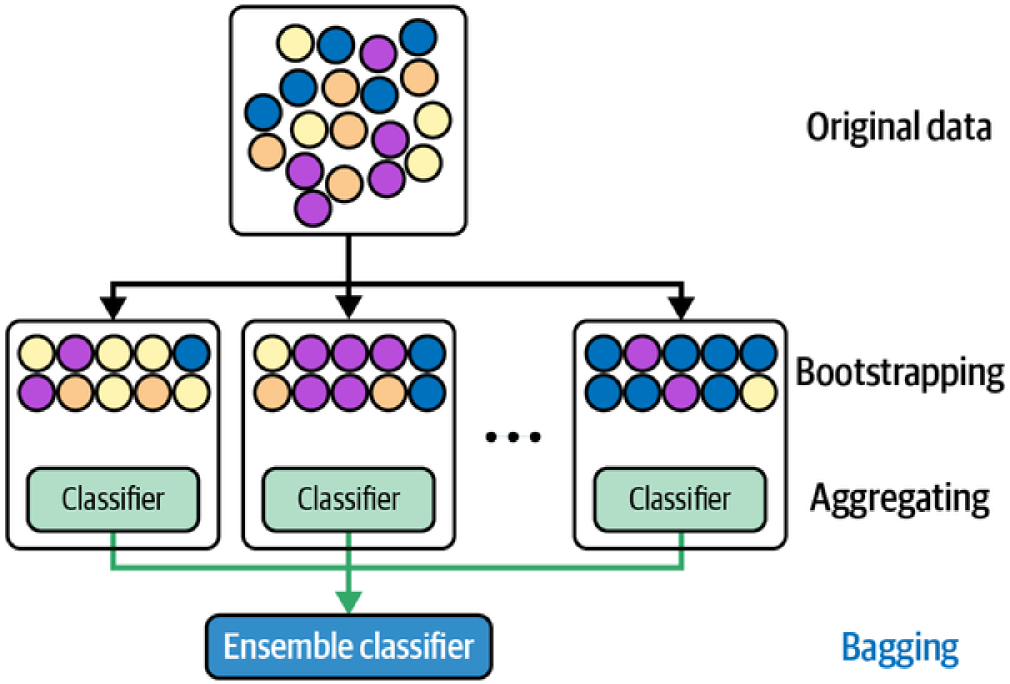
SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Jun 04, 2021	IE-Net (ensemble) *RICOH_SRCB_DML*	**90.939**	**93.214**
2 Feb 21, 2021	FPNet (ensemble) *Ant Service Intelligence Team*	90.871	93.183
3 May 16, 2021	IE-NetV2 (ensemble) *RICOH_SRCB_DML*	90.860	93.100
4 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	90.724	93.011
5 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
5 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.578	92.978
5 Feb 05, 2021	FPNet (ensemble) *YuYang*	90.600	92.899

Possible Outcomes

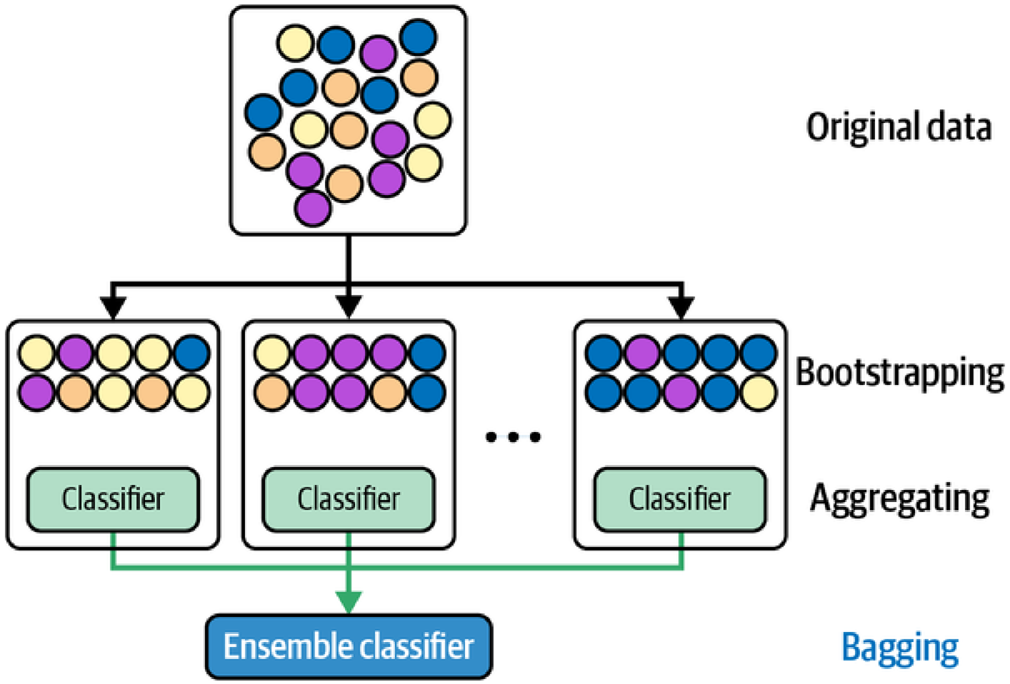
Outputs of three models	Probability	Ensemble's output
All three are correct	0.7 * 0.7 * 0.7 = 0.343	Correct
Only two are correct	(0.7 * 0.7 * 0.3) * 3 = 0.441	Correct
Only one is correct	(0.3 * 0.3 * 0.7) * 3 = 0.189	Wrong
None are correct	0.3 * 0.3 * 0.3 = 0.027	Wrong

Bagging

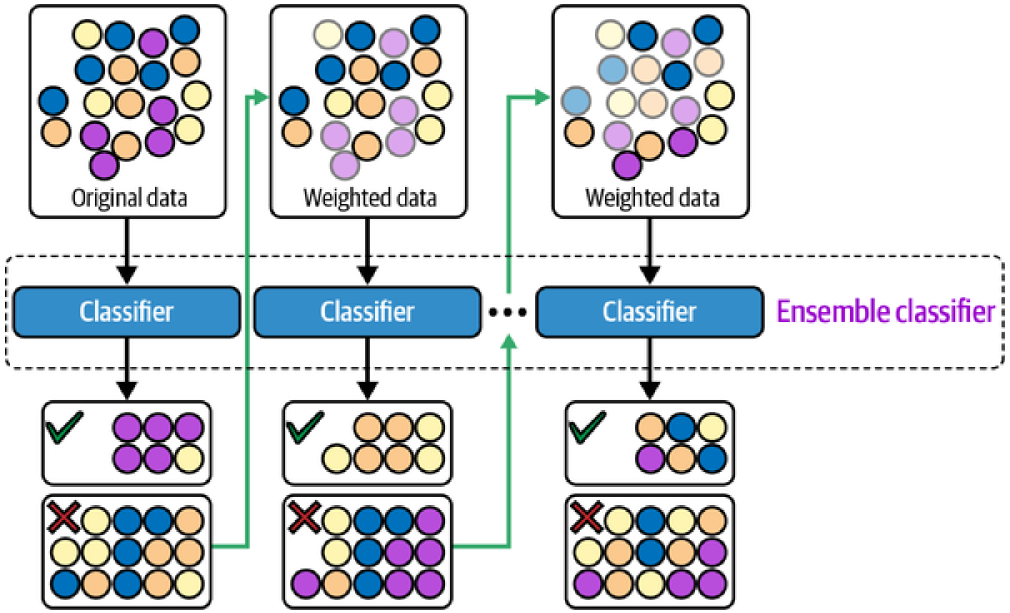


- Bagging (bootstrap aggregating) is designed to improve ML algorithms' training stability and accuracy.
- Reduces variance and helps avoid overfitting; it improves unstable methods (e.g., tree-based methods)
- Sampling with replacement ensures that each bootstrap is created independently from its peers.

Bagging

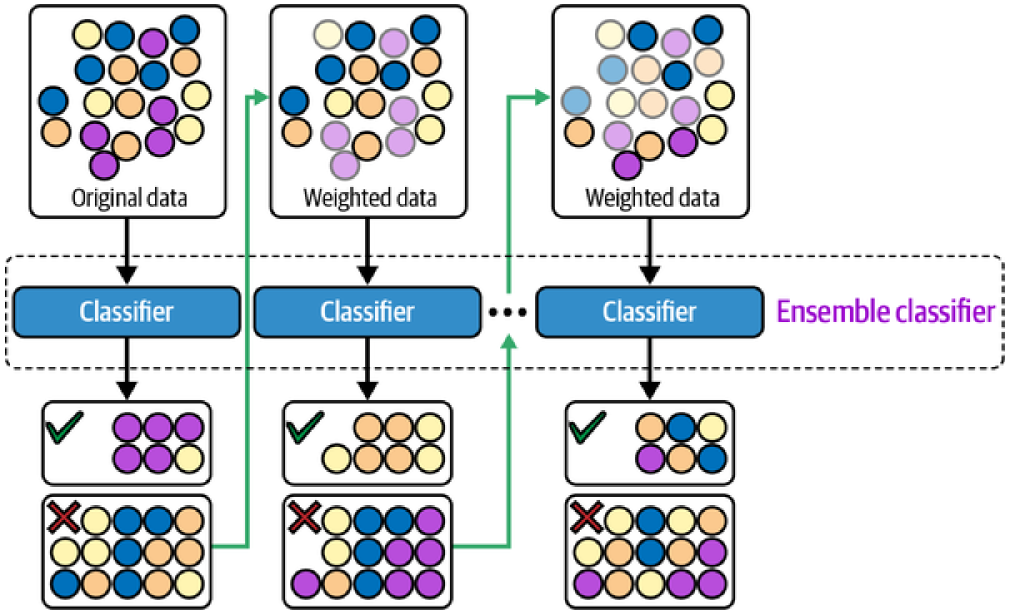


- Outline:
  - Given a data set, create n data sets by sampling with replacement (bootstrap).
  - Train classification or regression model on each bootstrap.
  - If classification, decide by majority vote; if regression, use the mean result.



Boosting

- Family of iterative ensemble algorithms that convert weak learners to strong ones.
- Examples: Gradient Boosting Machine (GBM), XGBoost, and LightGBM.

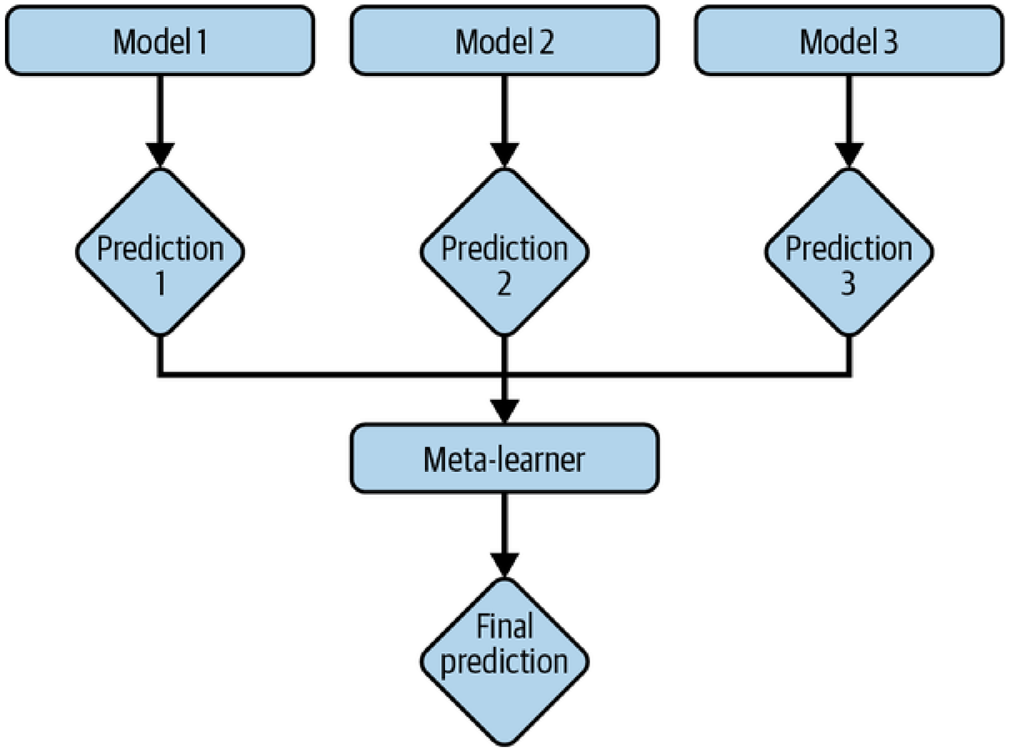


Boosting

- Outline:
  - Each learner is trained on the same set of samples, but the samples are weighted differently in each iteration.
  - Future weak learners focus more on the examples that previous weak learners misclassified.

Stacking

- Outline:
    - Create base learners from the training data.
    - Create a meta-learner that combines the outputs of the base learners to output predictions.



Experiment Tracking and Versioning

Experiment Tracking

- The process of tracking the progress and results of an experiment is called experiment tracking.
- ML Flow and Weights & Balances are experiment tracking tools.
- At a minimum, track performance (loss) and time (speed).
- Values over time of any parameter and hyperparameter whose changes can affect model performance.

Experiment Tracking

- Model performance metrics : on all nontest splits like accuracy, F1, perplexity.
- Loss curve: train split and each of the eval splits.
- Log of corresponding sample, prediction, and ground truth labels.
- Speed of the model: number of steps per second or tokens processed per second.
- System performance metrics: memory, CPU, GPU.

Versioning

- The process of logging an experiment's details to recreate it later or compare it with other experiments is called versioning.
- ML models in production are part code and part data.
- Code versioning has more or less become a standard in the industry.
- Data versioning is not standard.

Versioning

- Code versioning tools allow you to switch between versions of the codebase by keeping copies of all the old files. Data may be too large for duplication to be feasible.
- Code versioning tools allow several people to work on the same code simultaneously by replicating locally. Data may be too large, as well.
- What is a diff when versioning data? DVC, for example, only checks in changes in checksum.
- Compliance with GDPR may also be problematic if full history of data is kept.

Making Progress

Debugging: Why ML Models Fail

- Theoretical constraints: model assumptions are not met. For example, use a linear model when decision boundaries are not linear.
- Poor implementation: The model may be a good fit, but implementation has errors.
- Poor choice of hyperparameters: with the same model, one set of hyperparameters can give better results than others.

Debugging: Why ML Models Fail

- Data problems: noise and dirty data are everywhere. Also, poor implementation of data flows can induce data problems.
- Poor choice of features: Too many features may cause overfitting or data leakage. Too few features might lack predictive power to allow to make good predictions.
- Some debugging approaches:
  - Start simple and gradually add more components.
  - Overfit a single batch.
  - Set a random seed.

AutoML

- AutoML is the automatic process of finding ML algorithms to solve real-world problems.
- The most popular form of AutoML is hyperparameter tuning.
- Searching the Hyperparameter space can be time-consuming and resource-intensive.

Model Offline Evaluation

- Measure model performance before and after deployment.
- Evaluation methods should (ideally) be the same for models during development and production.
- Techniques for model offline evaluation:
  - Use baselines.
  - Tests: perturbation tests, invariance tests, directional expectation tests, model calibration, confidence measurement, slice-based evaluation.

Model Offline Evaluation: Baselines

- Random baseline: if the model predicts at random, how would it perform?
- Simple heuristic: how does the model perform vs a simple (non-ML) rule of thumb?
- Zero rule baseline: trivial prediction, always predicts the same thing.
- Human baseline: human-level performance may be the required baseline.
- Existing solutions.

Evaluation Methods in Production

- Perturbation tests: make changes to test splits, such as adding noise to input data. If a model is not robust to noise, it will be difficult to maintain.
- Invariance tests: specific input changes should not lead to output changes—for example, protected classes.
- Directional expectation tests.

Evaluation Methods in Production

- Model calibration or conformal prediction methods:
  - Idea: If the forecast has a 70% chance of rain, then 70% of the time this forecast was made, it actually rained.
  - Prediction scores are many times normalized to values between 0 and 1. It is tempting to think of them as probabilities, but they are not necessarily so.
  - Use conformal prediction methods to calibrate prediction scores.
- Confidence measurement: show only predictions where the model is confident.
- Slice-based evaluation: model performance is different in subsets of data.

References

References

- Agrawal, A. et al. "Cloudy with a high chance of DBMS: A 10-year prediction for Enterprise-Grade ML." arXiv preprint arXiv:1909.00084 (2019).
- Huyen, Chip. "Designing machine learning systems." O'Reilly Media, Inc.(2022).
- Tetlock and Gardner. Superforecasting: The art and science of prediction. Random House, 2016.