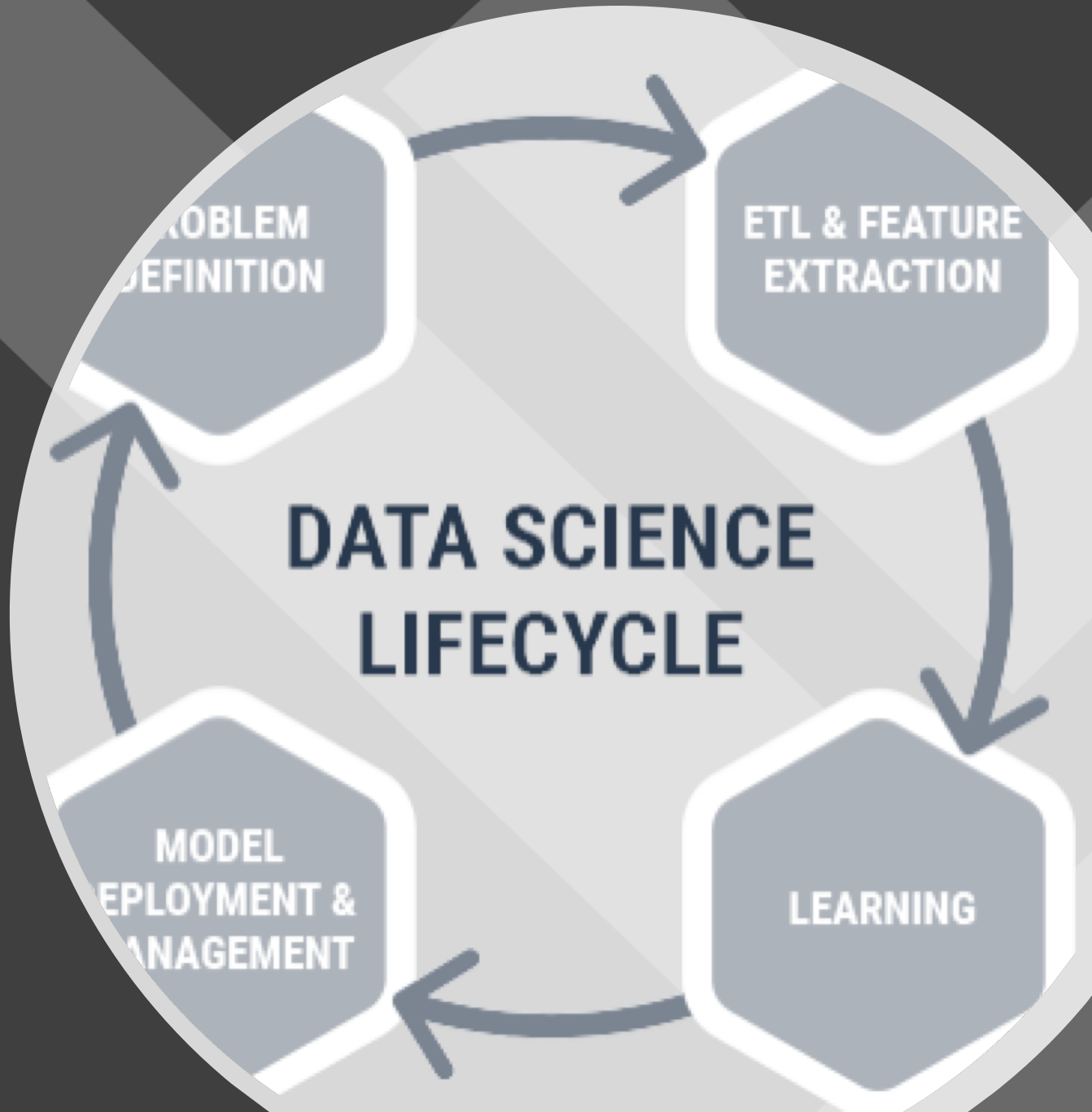
Overcoming the Development Challenges of Machine Learning Products

Ohad Zadok

Agenda

1. Machine learning project's life cycle recap
2. Case Study #1 – Multiclass Classifier
   a. *Don't #1* – Stack models horizontally
   b. *Do* : End-to-End training
   c. *Do* : Parallel Model Stacking
   d. DAM – Highly parallel architecture
3. Case Study #2 – Graph Traversal
   a. *Don't #2* - Introduce manual steps to automatic flows
   b. *Do :* DDD (Data Driven Development)
   c. *Do*: Single Function of Evaluation
   d. *Do:* Use Bayesian Optimization

The Lifecycle of a Machine-Learning Product



Problem Definition

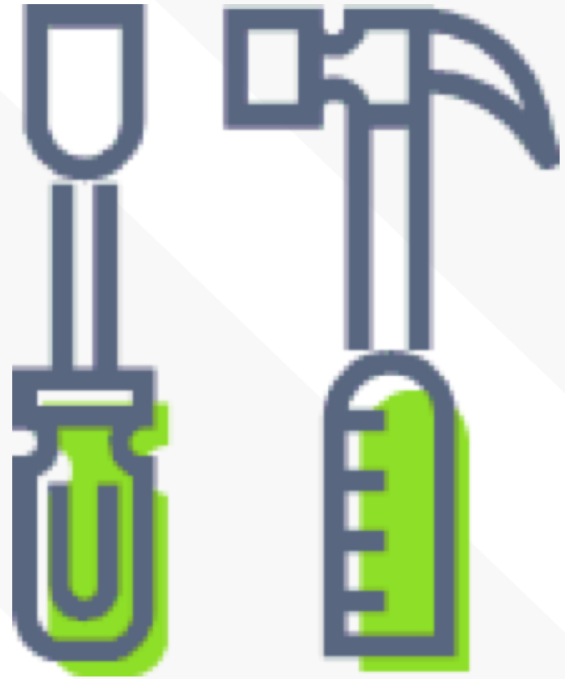
- Business definition of the problem; addressing a need
- Set a clear success criteria
- Similar approach to all software development projects

Collecting Data

- Get a dataset of examples of the problem you are trying to solve
  - Example: Phishing email detection – collect both safe email messages and phishing examples to kickoff the learning process
- Split data to ‘Train’ and ‘Test’ sets (some might even add a ‘Validation’ set)

Model Building

- Feature extraction - Transform the collected data into machine-readable format
- Test out different algorithms
- Find the highest scoring model



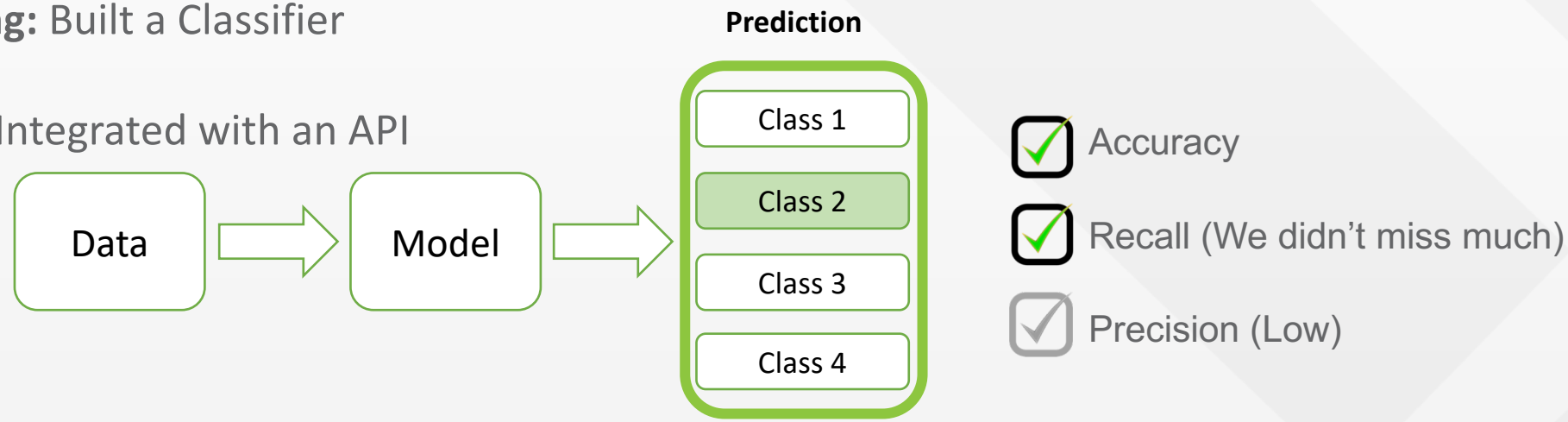
Deployment

- Pack it and serve it with an API
- Stress tests to handle load
- Create monitoring

Case Study #1 - Vertical Model Stacking

Case Study – Multiclass Classifier

- **Problem definition:** Create multiclass classifier with high Recall (our coverage had to be high)
- **Data Collection:** Hand tagged data.
- **Model Building:** Built a Classifier
- **Deployment:** Integrated with an API



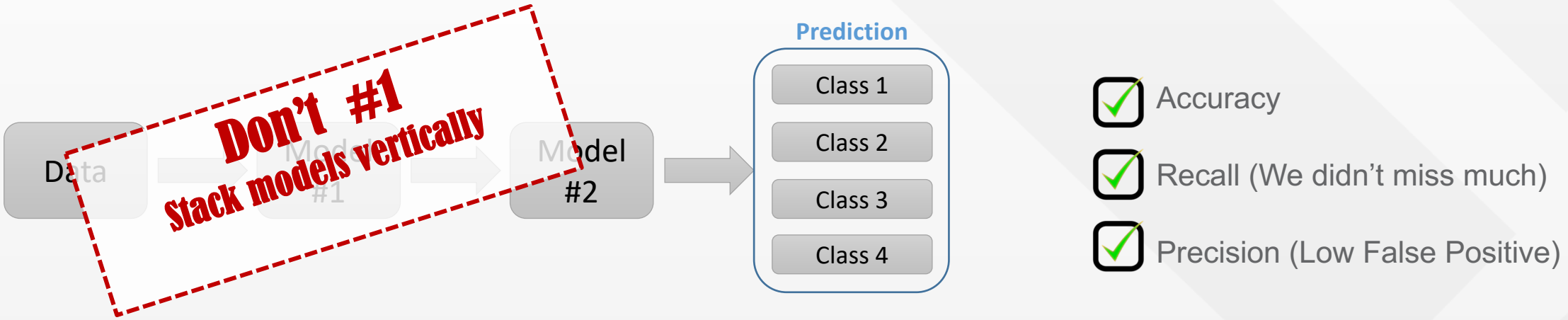
Case Study – Multiclass Classifier
Business definitions changed

- **Problem definition:** Create multiclass classifier with high Recall **and High Precision**
- **Data Collection:** Hand tagged data.
- **Model Building:** Naive Bayes **+ One more Classifier**
- **Deployment:** Integrated with an API

Case Study – Multiclass Classifier

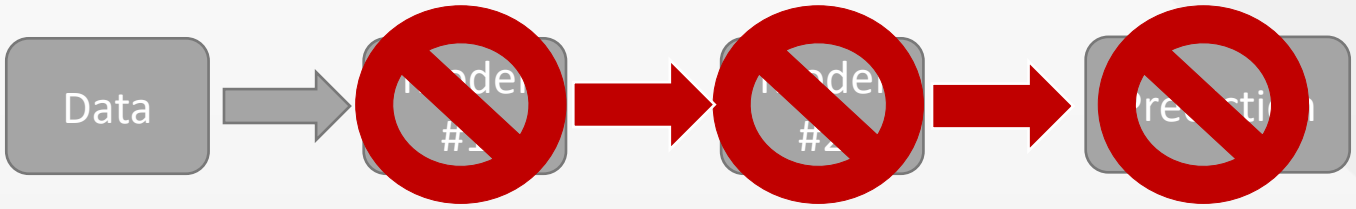
New Architecture

- We added vertical model to improve Precision
- Our hope was that the new model would pick up where the old one ended…



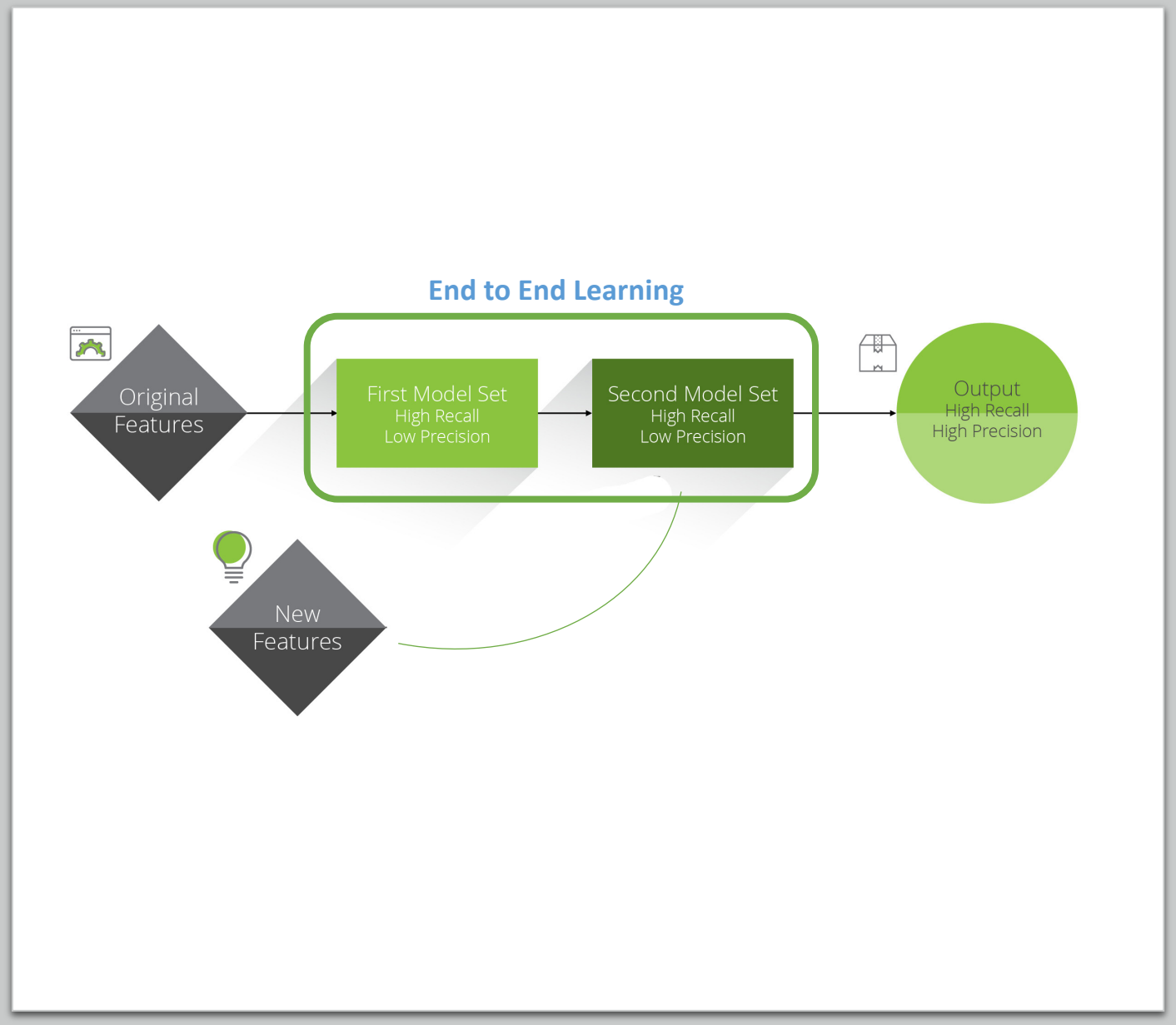
Don't #1: Vertical model stacking

- Domino effect - change to an early model in the stack had to propagate all the way to the end (Change to Model #1 – invalidates model #2)
- Cost of change increase greatly (from a few days to more than three weeks!)



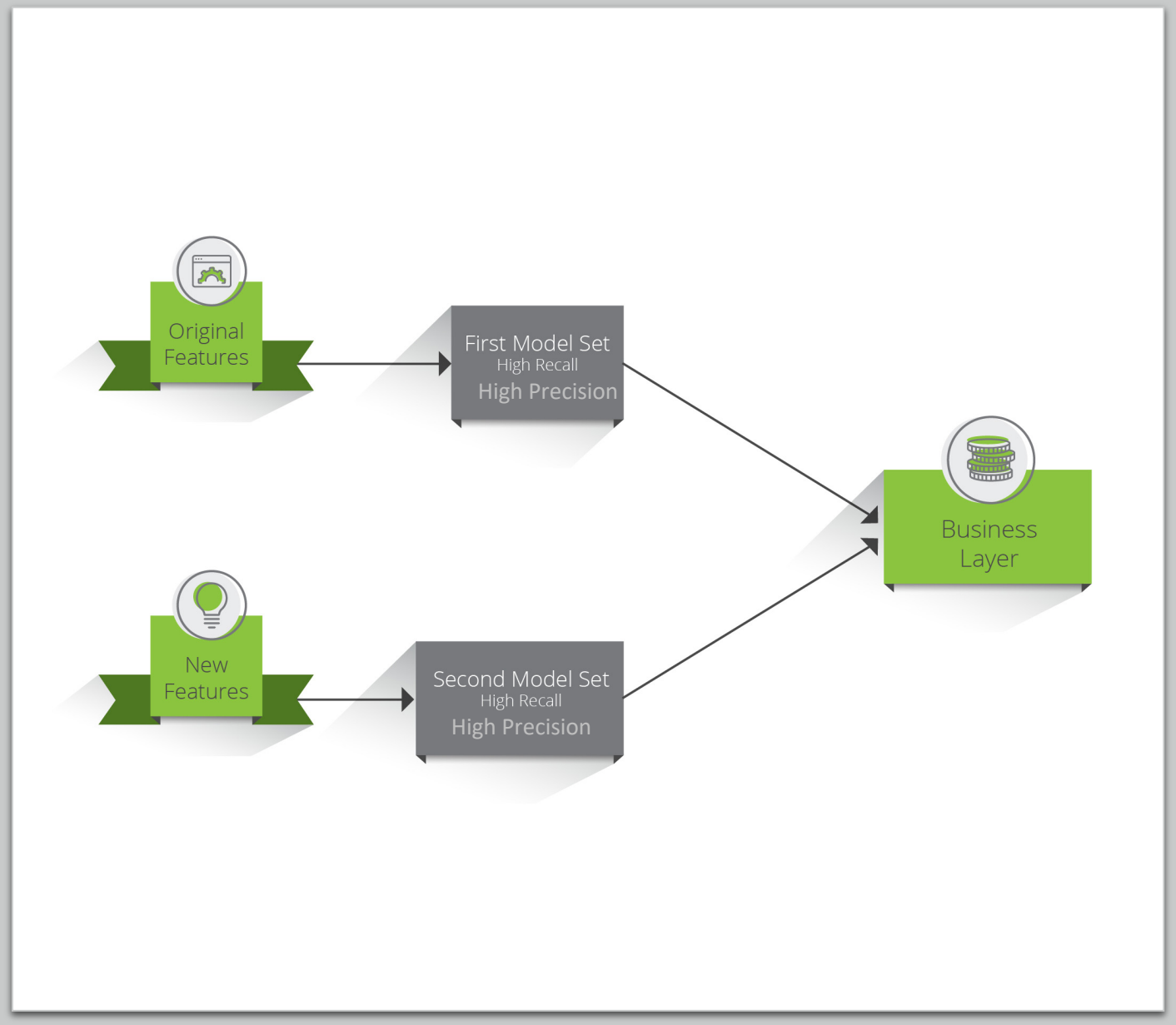
Do : End-to-End training

- ✓ Combine two models into one Automatic solution
- ✓ Train them together with a push of a button
- ✓ Example: Neural network with multiple layers – all layers are being trained together



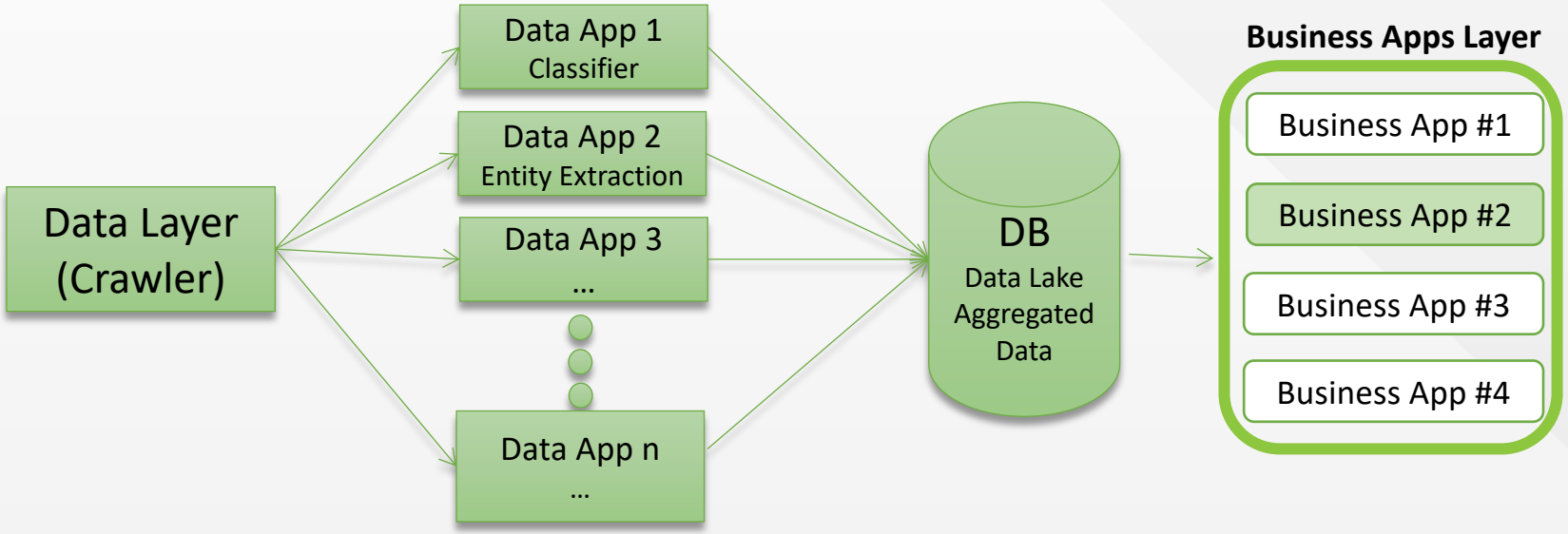
Do : Parallel Model Stacking

- ✓ Independent solutions - Can be updated solely
- ✓ Each solution meets quality standards



Architecture Suggestion - DAM

1. Easy to Deploy Apps (Micro services)
2. Deploy apps both in Beta and in Production maturity level
3. Filter by version (0.x for beta)
4. Enable Rapid development
   (Avoid current Production-Deployment Timelines)



Case Study #2 – Graph Traversal

Case Study #2 – Graph Traversal

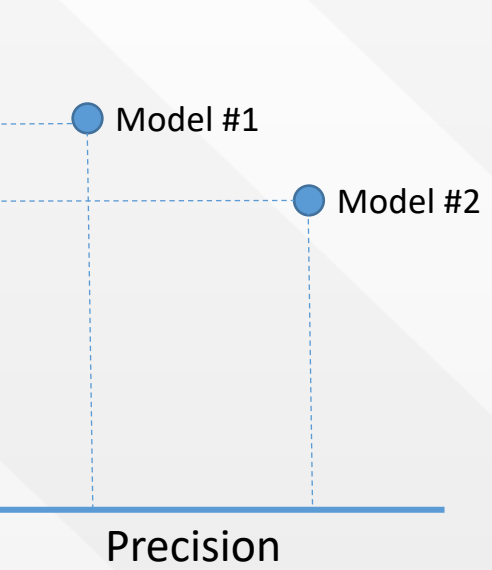
- **Problem definition:** Finding business affiliation between graph entities.

  Multiple factors for success:

  - Low number of results
  - High Precision
  - High Recall
- **Data:** No supervised data, only infinite Graph
- **Model Building:** Graph-Traversal algorithm
- **Deployment:** API

Case Study – Graph Traversal

- Problem: How to understand quality?
- Solution: Complex manual analysis – check if it serves the business needs
- Outcome:
  - Time per iteration took days of analysis – Inefficient!
  - Non-conclusive results (Which model is better?)



Do: DDD (Data Driven Development)

- Create a test set which resemble your problem before you start developing (Like TDD)
- Allows fast testing of solutions/hypothesis
- Allows usage of automatic algorithms for parameters tuning – like **Bayesian optimization**

Do: Single Function of Evaluation

- ✓ Can be as complex as you want
- ✓ Should return a single value, higher is better!
- ✓ Example: F1 combines recall and precision

$$\mathrm{F}_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Do: Use Bayesian Optimization

- Finds a minimum of a given function
- Function can be anything that return a scalar

```
1  from skopt import gp_minimize
2
3  def foo(x):
4      print x
5      return x[0]**4
6
7  res = gp_minimize(foo, [(range)])
```

Case Study – Final Solution

Complex business problem – multiply factors for assessments

- ✓ Created a Test set – took us a week
- ✓ Defined F1 as our evaluation function
- ✓ Used **Bayesian optimization** for parameters tuning
- ✓ Result: 30% score improvement over previous solutions (took us 1 hour end-to-end!!)
- ✓ Future development cycles will be much faster

EverCompliant
More Intelligence. Less Risk.

Thank You!

Check out our tech blog for more Do's and Don'ts

http://evercompliant.com/machine-learning-dos-donts-part1/