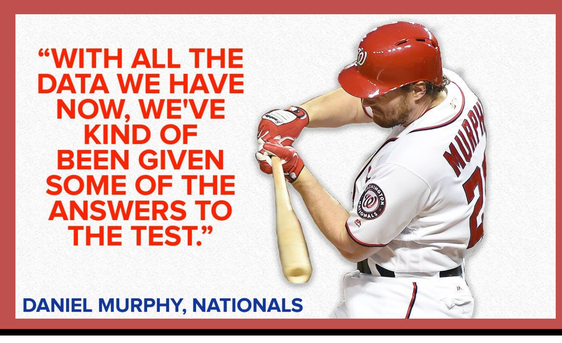
Towards the Realization of a DSML for Machine Learning: A Baseball Analytics Use Case

Kaan Koseler & Matthew Stephan
Miami University
{koselekt, stephamd} @miamioh.edu



Motivation

- Building quality Machine Learning (ML) software is challenging
  - Requires domain, ML, and software engineering knowledge
  - Difficult for organizations to find individuals possessing requisite knowledge
  - MDE holds promise in facilitating development of ML software
- Our goal: Demonstrate feasibility of applying MDE in ML domain through a baseball analytics use case

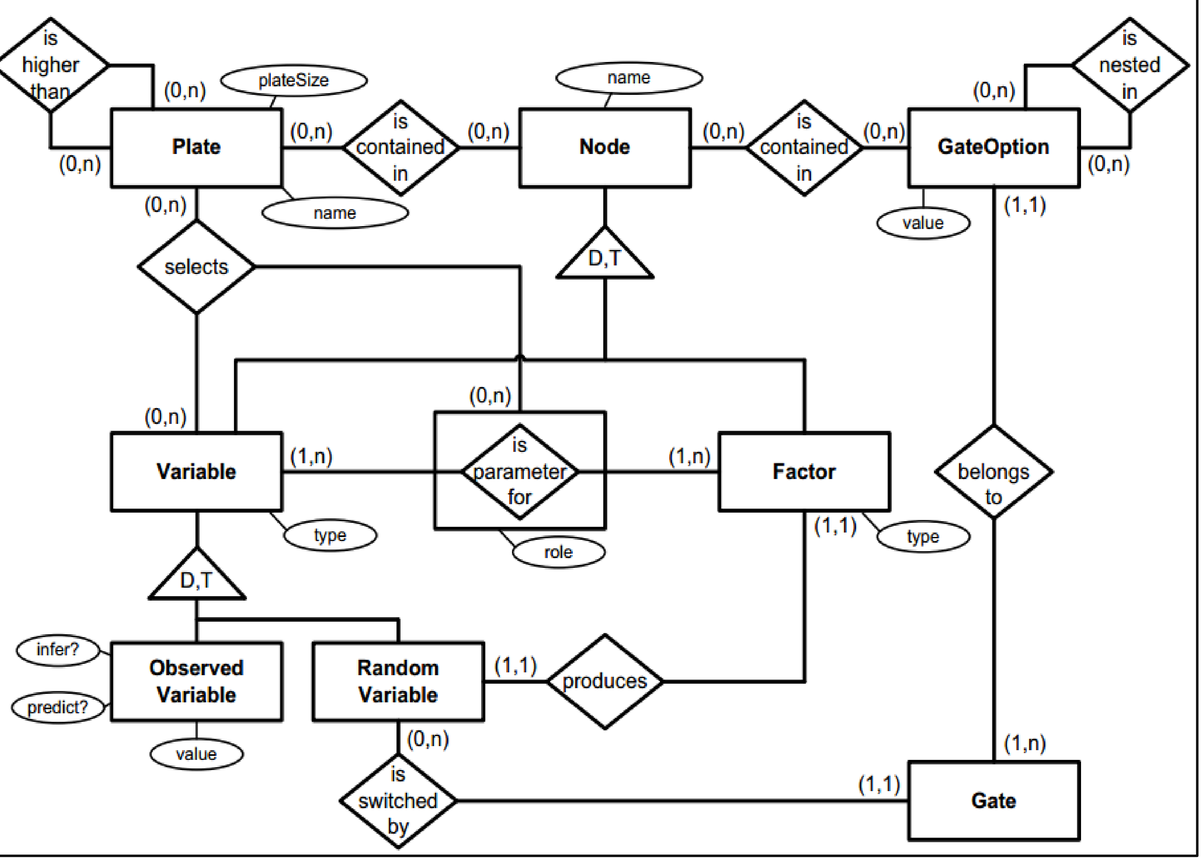
Details of Approach

- Plan on extending and applying proposed, but untested, DSML (Breuker, 2014) built on the Infer.NET C# library
- Implement code generation
- Apply it to a specific Baseball Analytics use case and compare against existing work
  - Binary classification problem: Predicting the next pitch in a baseball game (Fastball or Non-Fastball) (Ganeshapillai & Guttag, 2012)
- Construct several different model instances with 37 different features (Home/Away game, Handedness, etc.)

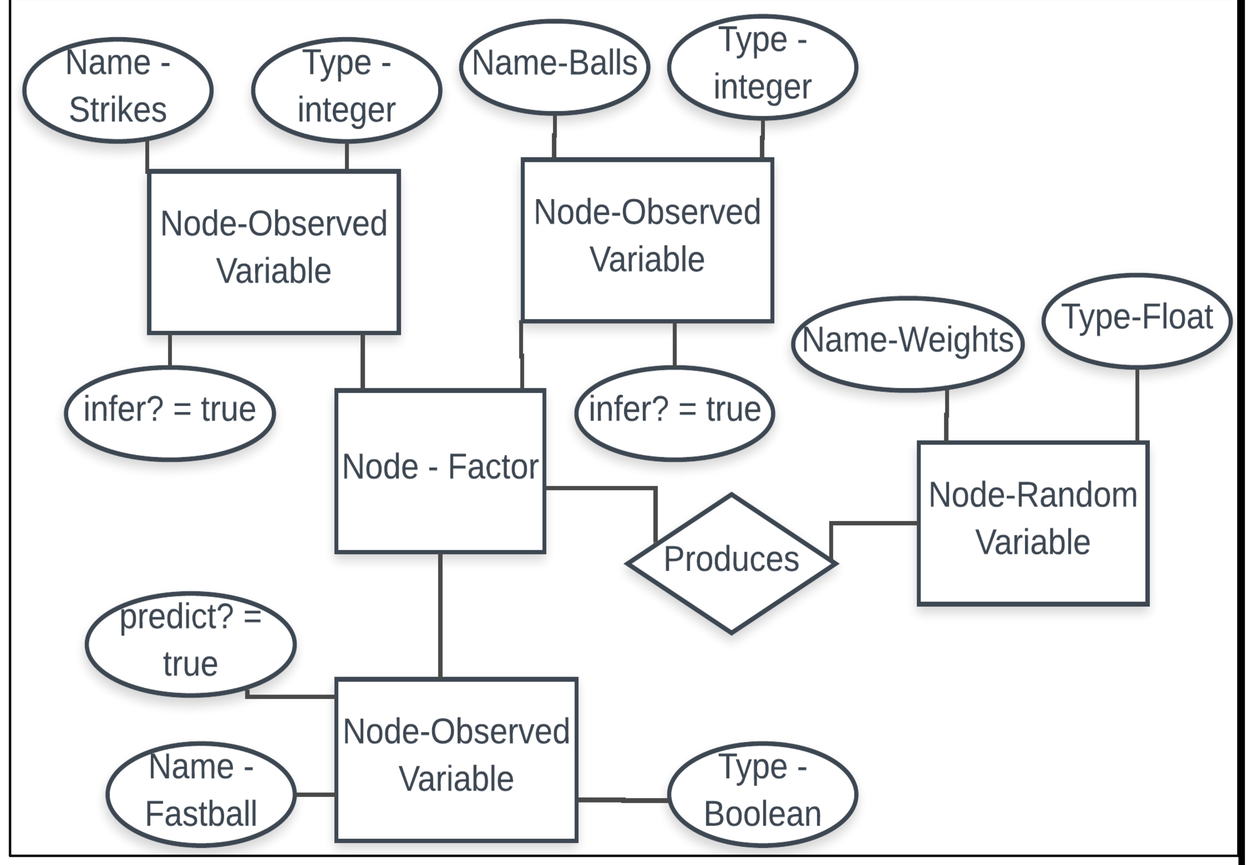
Initial Example

- Model instance uses count (# of balls and strikes) as a feature to predict the next pitch
- Contains three observed variables with a factor representing the relationship between them
- This relationship ultimately produces weights to be used for inference and prediction
- This DSM models 2 features, whereas our target DSM will model 37

Proposed DSML (Breuker 2014)



Sample Model Instance



Sample Training Code

```
// The labeled training data
double[] strikes = { 2, 1, 2, 0, 1, 0 };
double[] balls = { 3, 3, 0, 2, 1, 0 };
bool[] fastball = { true, false, true, true, false, false };

//create target vector and weight vector
VariableArray<bool> y = Variable.Observed(fastball).Named("y");
Variable<Vector> w = Variable.Random(new VectorGaussian(Vector.Zero(2),
    PositiveDefiniteMatrix.Identity(2))).Named("w");

    //Call the bayes point machine method
    BayesPointMachine(strikes, balls, w, y);
}

public void BayesPointMachine(double[] strikes, double[] balls, Variable<Vector> w, VariableArray<bool> y)
{
    // Create training data vector
    Range j = y.Range;
    Vector[] xVector = new Vector[balls.Length];
    for (int i = 0; i < xVector.Length; i++)
        xVector[i] = Vector.FromArray(strikes[i], balls[i]);
    VariableArray<Vector> x = Variable.Observed(xVector, j).Named("x");

    // Bayes Point Machine, dot product of weights and feature vector
    y[j] = Variable.GaussianFromMeanAndVariance(Variable.InnerProduct(w, x[j]).Named("dotProduct")) > 0;
}
```

Sample Inference Code

```
//Create inference engine object and infer
//posterior distribution of weights
InferenceEngine engine = new InferenceEngine();
VectorGaussian wPosterior = engine.Infer<VectorGaussian>(w);
Console.WriteLine("Dist over w=\n" + wPosterior);

//Make predictions on test data
double[] strikesTest = { 2, 1, 2 };
double[] ballsTest = { 3, 2, 0 };
VariableArray<bool> ytest = Variable.Array<bool>(new Range(strikesTest.Length)).Named("ytest");
BayesPointMachine(strikesTest, ballsTest, Variable.Random(wPosterior).Named("w"), ytest);
Console.WriteLine("output=\n" + engine.Infer(ytest));
```

References

•Breuker, D.: Towards model-driven engineering for big data analytics–an exploratory analysis of domain-specific languages for machine learning. In: Hawaii International Conference on System Sciences. pp. 758–767 (2014)
•Ganeshapillai, G., Guttag, J.: Predicting the next pitch. In: Sloan Sports Analytics Conference (2012)

Acknowledgements

This work is supported by the Miami University Senate Committee on Faculty Research (CFR) Faculty Research Grants Program