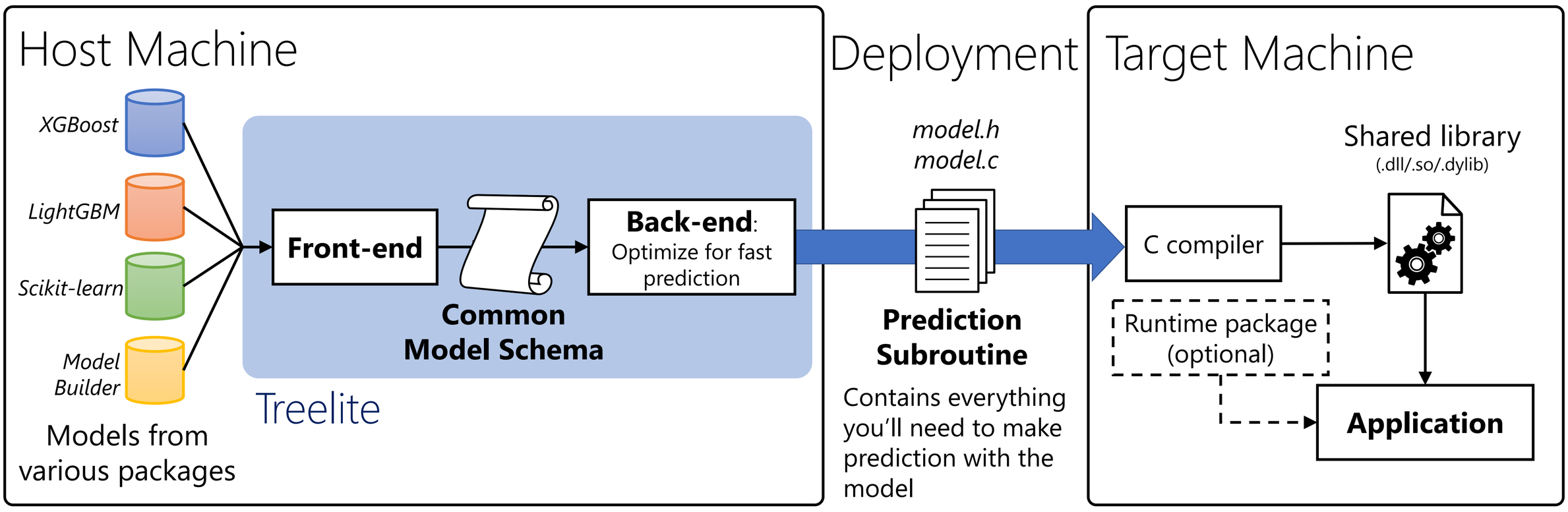
Treelite: toolbox for decision tree deployment

Hyunsu Cho, University of Washington
Mu Li, Amazon Web Services



Why Treelite?

Provide common interface for predicting with multiple tree models and libraries

- Support multiple tree models: Random Forests vs. Gradient Boosted Trees
- Support multiple tree libraries: XGBoost, LightGBM, Scikit-learn etc.
- Deploy prediction logic apart from other dependencies

Speed up prediction by compiling trees

- Prediction throughput is particularly important for batch prediction tasks (e.g. ad ranking, doc scoring)
- Provide GCC/Clang with model information at compile-time, allowing for better optimization

Convenient API

```
from treelite import Model
from treelite.runtime import Predictor
model = Model.load('my.model', 'xgboost')
model.export_lib('gcc', './mymodel.so')
predictor = Predictor('./mymodel.so')
```

Compiling decision trees

```
if ( /* comparison test for the test node */ ) {
  /* ... code for the left child node ... */
} else {
  /* ... code for the right child node ... */
}
```

Resources

- Repository: https://github.com/dmlc/treelite
- Documentation: http://treelite.io

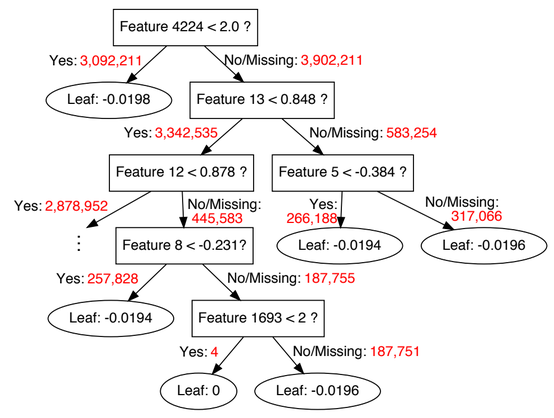
Acknowledgements

Treelite builds upon Hyunsu's earlier work at University of Washington, which was performed under the guidance of **Carlos Guestrin** and **Arvind Krishnamurthy**. **Tianqi Chen**, a student of the same institution, offered many great ideas for API design.

Back-end Optimization

Annotate branches

Tell GCC/Clang the expected likelihood of each split condition



Map split thresholds to integers

More to be added in the future

Benchmarks

- One EC2 instance, of type `m4.16xlarge`
- Gradient boosted trees, 1600 trees
- **2-5x speedup**
- Optimizations **cuts CPI by half**