

Scikit Model inference in C++

Abhilash Babu

- Using models trained using scikit-learn in a C++ Application
 - Available options
 - ONNX
 - Treelite
 - PMML
 - ...
- Demo

Available Options

- Use an intermediate format
 - From the scikit-learn documentation
 - ONNX
 - PMML
- Use the same underlying library that scikit learn uses
 - liblinear
 - libsvm
- Other options
 - treelite

- Stands for **Open Neural Network Exchange**
- Definition from the official ONNX website
 - ONNX is an open format built to represent machine learning models.
 - ONNX defines a common set of operators - the building blocks of machine learning and deep learning models.
 - ONNX also defines a common file format to enable AI developers to use models with a variety of frameworks, tools, runtimes, and compilers.

```
//LOAD MODEL
Ort::MemoryInfo info("Cpu", OrtDeviceAllocator, 0, OrtMemTypeDefault);
auto input_tensor = Ort::Value::CreateTensor<float>(info, const_cast<float*>(input.data()),
                                                    input.size(),
                                                    _input_shape.data(),
                                                    _input_shape.size());

// RUN INFERENCE
auto ort_outputs = _session.Run(Ort::RunOptions{ nullptr },
                                _input_names.data(),
                                &input_tensor, 1,
                                _output_names.data(), 2);

// GET OUTPUT
auto type_info = ort_outputs[0].GetTensorTypeAndShapeInfo();
auto data_length = ort_outputs[0].GetStringTensorDataLength();
std::string result(data_length, '\0');
std::vector<size_t> offsets(type_info.GetElementCount());
ort_outputs[0].GetStringTensorContent((void*)result.data(), data_length, offsets.data(),
                                       offsets.size());
```

Figure 1: Screenshot

Why ONNX

- Open source.
- Community project backed by top companies
- Works with models of a wide range of frameworks.
- Works on a variety of platforms

Optimize Inference	Optimize Training							
Platform	Windows	Linux	Mac	Android	iOS	Web Browser (Preview)		
API	Python	C++	C#	C	Java	JS	Obj C	Wikit
Architecture	x86		x86		ARMv8-A		ARMv7	
Hardware Acceleration	Default CPU		CUDA		DirectML		oneDNN	
	TensorRT		MNNAPI		ACL (Preview)		ArmNN (Preview)	
	MIGraphX (Preview)		NUPHAR (Preview)		BackShip NPU (Preview)		Vitis AI (Preview)	
Installation Instructions	Install Next package: Microsoft.ML.OnnxRuntime							

CEPR is supported by a community of partner



Use the frameworks you already know and love.

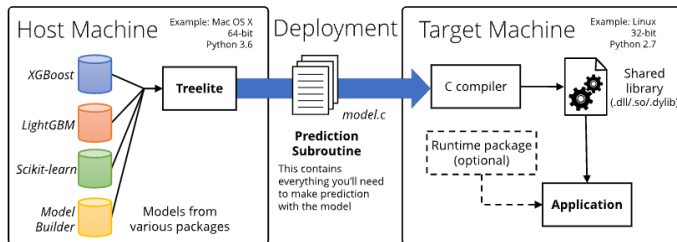
Use the frameworks you already know and love.



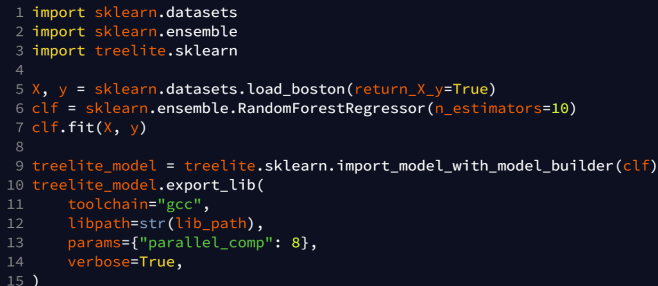
- Stands for Predictive Model Markup Language.
- XML based predictive model interchange format.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PMML xmlns="http://www.dmg.org/PMML-4_4" xmlns:data="http://jpmml.org/jpmml-model/InlineTable" version="4.4">
  <Header>
    <Application name="JPMML-SkLearn" version="1.6.31"/>
    <Timestamp>2021-10-24T03:30:59Z</Timestamp>
  </Header>
  <MiningBuildTask>
    <Extension name="repr">PMMLPipeline(steps=[('classifier', RandomForestClassifier(n_estimators=20, random_state=0))])</Extension>
  </MiningBuildTask>
  <DataDictionary>
    <DataDictionary>
      <MiningModel functionName="classification" algorithmName="sklearn.ensemble._forest.RandomForestClassifier">
        <MiningSchema>
          <MiningSchema>
            <OutputField name="probability(0)" optype="continuous" dataType="double" feature="probability" value="0"/>
            <OutputField name="probability(1)" optype="continuous" dataType="double" feature="probability" value="1"/>
            <OutputField name="probability(2)" optype="continuous" dataType="double" feature="probability" value="2"/>
            <OutputField name="probability(3)" optype="continuous" dataType="double" feature="probability" value="3"/>
            <OutputField name="probability(4)" optype="continuous" dataType="double" feature="probability" value="4"/>
            <OutputField name="probability(5)" optype="continuous" dataType="double" feature="probability" value="5"/>
            <OutputField name="probability(6)" optype="continuous" dataType="double" feature="probability" value="6"/>
            <OutputField name="probability(7)" optype="continuous" dataType="double" feature="probability" value="7"/>
            <OutputField name="probability(8)" optype="continuous" dataType="double" feature="probability" value="8"/>
            <OutputField name="probability(9)" optype="continuous" dataType="double" feature="probability" value="9"/>
          </Output>
          <LocalTransformations>
          </LocalTransformations>
          <Segmentation multipleModelMethod="average">
          </Segmentation>
        </MiningModel>
      </DataDictionary>
    </DataDictionary>
  </MiningModel>
</PMML>
```

- Definition from the website
 - **Treelite is a model compiler for decision tree ensembles, aimed at efficient deployment.**
 - Treelite overview



Treelite Example



```
1 import sklearn.datasets
2 import sklearn.ensemble
3 import treelite.sklearn
4
5 X, y = sklearn.datasets.load_boston(return_X_y=True)
6 clf = sklearn.ensemble.RandomForestRegressor(n_estimators=10)
7 clf.fit(X, y)
8
9 treelite_model = treelite.sklearn.import_model_with_model_builder(clf)
10 treelite_model.export_lib(
11     toolchain="gcc",
12     libpath=str(lib_path),
13     params={"parallel_comp": 8},
14     verbose=True,
15 )
```

Figure 2: Treelite code

- https://github.com/abhilb/pydata_2021
- <https://onnxruntime.ai/>
- <https://onnx.ai/>
- <https://treelite.readthedocs.io/en/latest/>
- <http://dmg.org/pmml/v4-4-1/GeneralStructure.html>