Decision Tree

In this series of tutorial, we delve into the world of traditional machine learning models for ZKML. Despite the hype surrounding advanced AI techniques, traditional ML models often offer superior performance or sufficiently robust results for specific applications. This is particularly true for ZKML use cases, where computational proof costs can be a critical factor. Our aim is to equip you with guides on how to implement machine learning algorithms suitable for Giza platform applications. This includes practical steps for converting your scikit-learn models to the ONNX format, transpiling them to Orion Cairo, and deploying inference endpoints for prediction in AI Action.

In this tutorial you will learn how to use the Giza tools though a Decision Tree model.

Before Starting

Before we start, ensure you installed the Giza stack, created a user and logged-in.

```

Copy !pipxinstallgiza-cli# Install the Giza-CLI !pipinstallgiza-actions# Install the AI Actions SDK

!gizauserscreate# Create a user !gizauserslogin# Login to your account !gizauserscreate-api-key# Create an API key. We recommend you do this so you don't have to reconnect.

```

Create and Train a Decision Tree Model

We'll start by creating a simple decision tree model using Scikit-Learn and train on iris dataset. We will then useHummingbirds library to convert the model to torch graphs.

```

Copy importjson importnumpyasnp fromsklearn.datasetsimportload_iris fromsklearn.model_selectionimporttrain_test_split fromsklearn.treeimportDecisionTreeClassifierasDe fromhummingbird.mlimportconvert importtorch importos

iris=load_iris() X,y=iris.data,iris.target X=X.astype(np.float32) X_train,X_test,y_train,y_test=train_test_split(X, y) clr=De() clr.fit(X_train, y_train)

model=convert(clr,"torch", X_test[:1]).model

```

Convert the Model to ONNX Format

Giza only supports ONNX models so you'll need to convert the model to ONNX format. After the model is trained, you can convert it to ONNX format.

```

Copy importtorch.onnx

input_sample=torch.from_numpy(X_test[:1])

# Specify the path to save the ONNX model

onnx_model_path="decision_tree.onnx"

# Export the model

torch.onnx.export(model, input_sample, onnx_model_path,# where to save the model export_params=True,# store the trained parameter weights inside the model file opset_version=17,# the ONNX version to export the model to do_constant_folding=True,# whether to execute constant folding for optimization input_names=['input'],# the model's input names output_names=['output'],# the model's output names dynamic_axes={'input': {0:'batch_size'},# variable length axes 'output': {0:'batch_size'}})

```

Transpile your model to Orion Cairo

We will use Giza-CLI to transpile our ONNX model to Orion Cairo.

```

Copy !gizatranspiledecision_tree.onnx--output-pathverifiable_dt

    [giza][2024-04-0316:03:36.422] No model id provided, checkingifmodelexists✅
    [giza][2024-04-0316:03:36.425] Model name is: decision_tree [giza][2024-04-
    0316:04:09.291] Model already exists, using existing model ✅ [giza][2024-04-
    0316:04:09.292] Model found with id ->146!✅ [giza][2024-04-0316:04:09.951]
    Version Created with id ->2!✅ [giza][2024-04-0316:04:09.952] Sending
    modelfortranspilation ✅ [giza][2024-04-0316:04:51.488] Transpilation is fully
    compatible. Version compiled and Sierra is saved at Giza ✅
    ⠸ TranspilingModel... [giza][2024-04-0316:04:52.201] Downloading model ✅
    [giza][2024-04-0316:04:52.241] model saved at: verifiable_dt

```

Deploy an inference endpoint

Now that our model is transpiled to Cairo we can deploy an endpoint to run verifiable inferences. We will use Giza CLI again to run deploy an endpoint. Ensure to replacemodel-id andversion-id with your ids provided during transpilation.

```

Copy !gizaendpointsdeploy--model-id146--version-id2

    ▰▰▱▱▱▱▱Creatingendpoint!t! [giza][2024-04-0316:05:20.152] Endpoint is
    successful ✅ [giza][2024-04-0316:05:20.156] Endpoint created with id ->36 ✅
    [giza][2024-04-03 16:05:20.156] Endpoint created with endpoint URL:
    https://endpoint-raphael-doukhan-146-2-98feddf7-6nn4ryaqca-ew.a.run.app

```

Run a verifiable inference in AI Actions

To streamline verifiable inference, you might consider using the endpoint URL obtained after transpilation. However, this approach requires manual serialization of the input for the Cairo program and handling the deserialization process. To make this process more user-friendly and keep you within a Python environment, we've introduced AI Actions—a Python SDK designed to facilitate the creation of ML workflows and execution of verifiable predictions. When you initiate a prediction, our system automatically retrieves the endpoint URL you deployed earlier, converts your input into Cairo-compatible format, executes the prediction, and then converts the output back into a numpy object. More info aboutAI Actions here.

First ensure you have an AI Actions workspace created. This step grants access to a user-friendly UI dashboard, enabling you to monitor and manage workflows with ease.

```

Copy !gizaworkspacesget

# If you haven't set up a workspace yet, you can establish one by executing the command below:

## ! giza workspaces create

    [giza][2024-04-0311:05:16.115] Retrieving workspace information ✅ [giza]
    [2024-04-0311:05:16.272] ✅ Workspace URL: https://actions-server-raphael-
    doukhan-dblzzhtf5q-ew.a.run.app ✅ { "url":"https://actions-server-raphael-
    doukhan-dblzzhtf5q-ew.a.run.app", "status":"COMPLETED" }

```

Now let's run a verifiable inference with AI Actions. To design your workflow in AI Actions, you will need to define your task with@task decorator and then action your tasks with@action decorator. You can track the progress of your workflow via the workspace URL previously provided.

```
```

Copy fromgiza_actions.modelimportGizaModel fromgiza_actions.actionimportaction fromgiza_actions.taskimporttask

MODEL_ID=146# Update with your model ID VERSION_ID=2# Update with your version ID

@task(name="PredictDTModel") defprediction(input,model_id,version_id): model=GizaModel(id=model_id, version=version_id)

(result,proof_id)=model.predict(input_feed={'input':input}, verifiable=True, custom_output_dtype="(Tensor, Tensor)"# Decision Tree will always have this output dtype.)

returnresult,proof_id

@action(name="ExectuteCairoDT", log_prints=True) defexecution():

The input data type should match the model's expected input

input=input_sample.numpy()

(result,proof_id)=prediction(input, MODEL_ID, VERSION_ID)

returnresult,proof_id

execution()

```

```

Copy 17:13:50.018|INFO|Createdflowrun'pistachio-polecat'forflow'ExectuteCairoDT' 17:13:50.019 | INFO | Action run 'pistachio-polecat' - View at https://actions-server-raphael-doukhan-dblzzhtf5q-ew.a.run.app/flow-runs/flow-run/0a1da6ea-a81a-4976-b605-28c7e425e16b 17:13:50.294|INFO|Actionrun'pistachio-polecat'-Createdtaskrun'PredictDTModel-0'fortask'PredictDTModel' 17:13:50.297|INFO|Actionrun'pistachio-polecat'-Executing'PredictDTModel-0'immediately... Startingdeserializationprocess... ✅Deserializationcompleted! 17:13:54.384|INFO|Taskrun'PredictDTModel-0'-FinishedinstateCompleted() 17:13:54.514|INFO|Actionrun'pistachio-polecat'-FinishedinstateCompleted() ((array([0]),array([[1.,0.,0.]])),'3bb53193c43048b7b47abfefe32b569a')

```

Download the proof

Initiating a verifiable inference sets off a proving job on our server, sparing you the complexities of installing and configuring the prover yourself. Upon completion, you can download your proof.

First, let's check the status of the proving job to ensure that it has been completed.

Remember to substituteendpoint-id andproof-id with the specific IDs assigned to you throughout this tutorial.

```

Copy !gizaendpointsget-proof--endpoint-id36--proof-id3bb53193c43048b7b47abfefe32b569a

[giza][2024-03-1911:51:45.470] Getting proof from endpoint 109 ✅ { "id":664, "job_id":831, "metrics":{ "proving_time":15.083126 }, "created_date":"2024-03-19T10:41:11.120310" }

```

Once the proof is ready, you can download it.

```

Copy !gizaendpointsdownload-proof--endpoint-id36--proof-id3bb53193c43048b7b47abfefe32b569a--output-pathzkdt.proof

[giza][2024-03-1911:55:49.713] Getting proof from endpoint 109 ✅ [giza][2024-03-1911:55:50.493] Proof downloaded to zklr.proof ✅

```
```

[Previous Linear Regression] [Next Build a Verifiable Neural Network with Giza Actions]

Last updated16 days ago