



# Machine Learning for Columnar High Energy Physics Analysis

**Elliott Kauffman** - Princeton University

Alexander Held - University of Wisconsin-Madison

Oksana Shadura - University of Nebraska-Lincoln



# Outline

**1. Analysis Grand Challenge Overview**

**2. Strategy for Integrating Machine Learning**

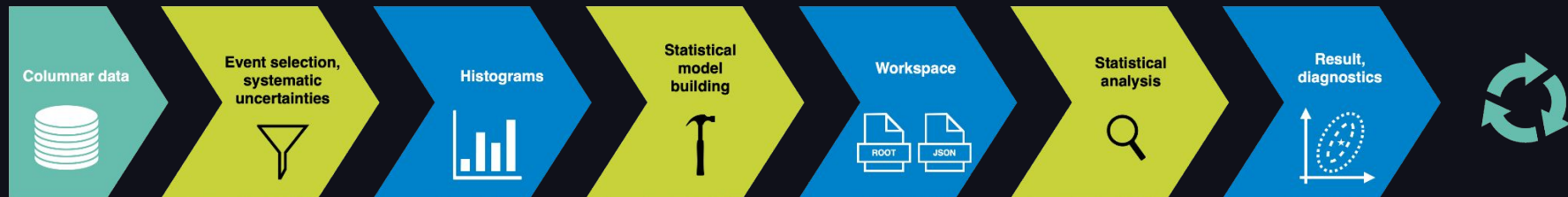
**3. Training Pipeline**

**4. Inference Pipeline**



# Analysis Grand Challenge

- Demonstrate **realistic analysis workflow** in anticipation of HL-LHC requirements
- Task: ttbar cross-section measurement using **2015 CMS Open Data**
  - ◆ Interactive and accessible user interface
  - ◆ **Open Data** - everyone can participate!
- Include all aspects of typical LHC analysis
  - ◆ Data access
  - ◆ Event selection
  - ◆ Histogramming
  - ◆ Statistical model building / fitting
  - ◆ Analysis preservation





→ What is needed for HL-LHC?

- ◆ Improvement in **algorithm performance**
- ◆ Better utilization of **computational resources**

→ Machine learning (especially deep learning) can help with this!

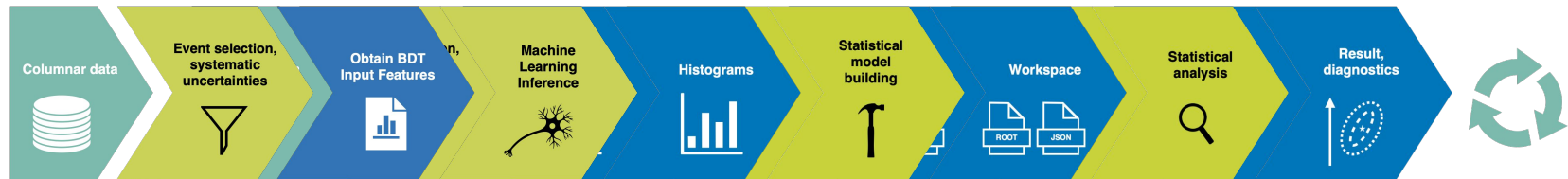
→ Add ML task to AGC pipeline to reflect developments in HEP analysis

# Machine Learning in HEP Analysis



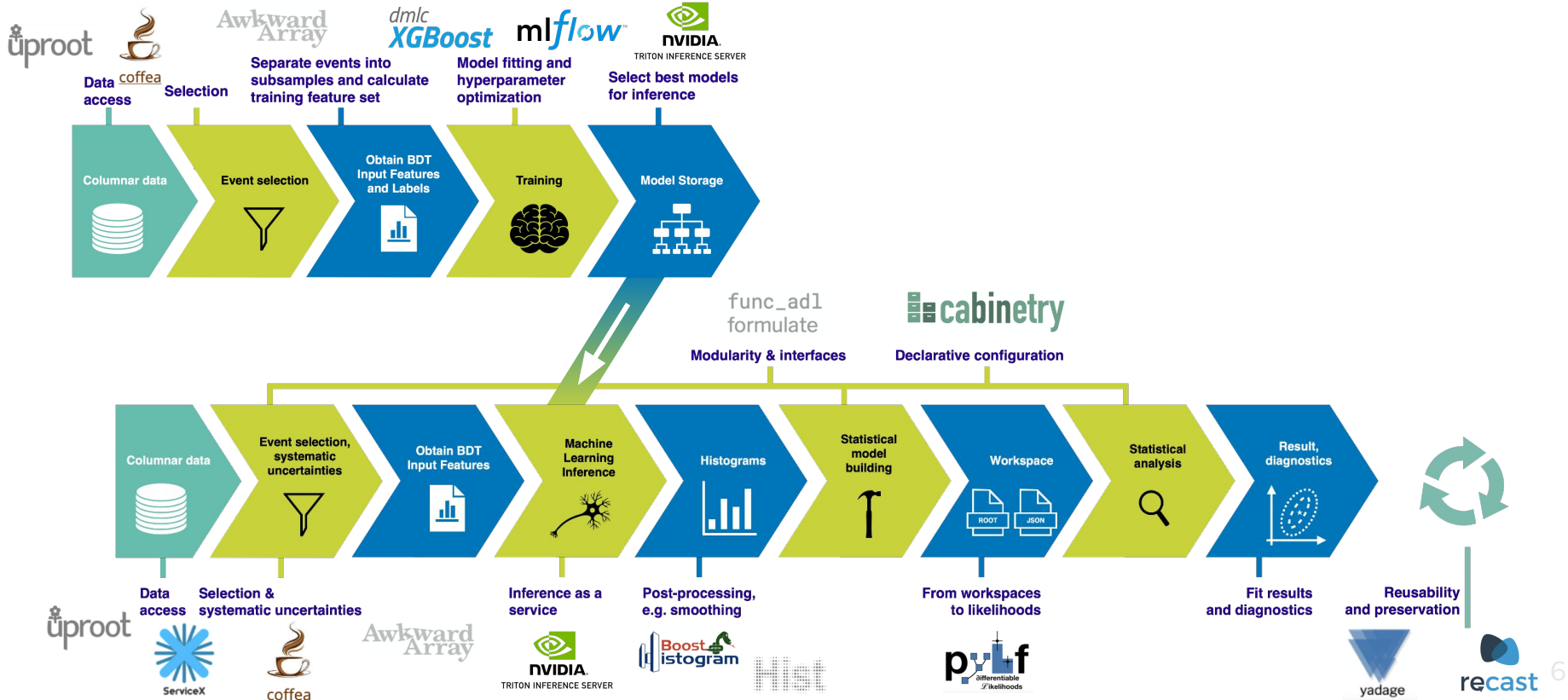


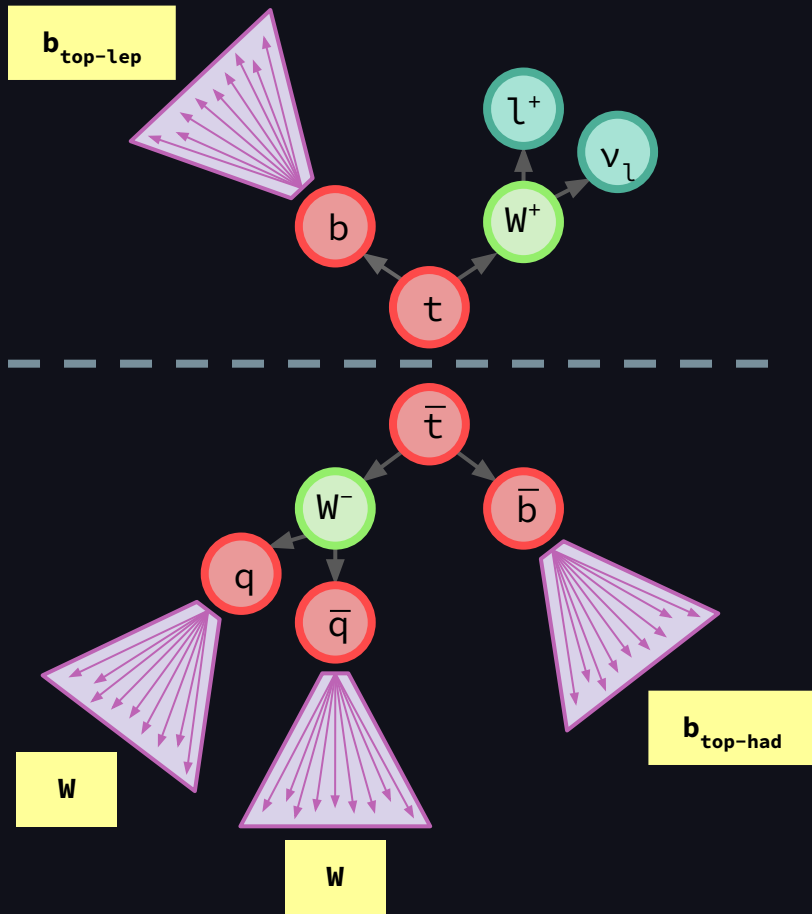
# Analysis Grand Challenge + ML





# Analysis Grand Challenge + ML





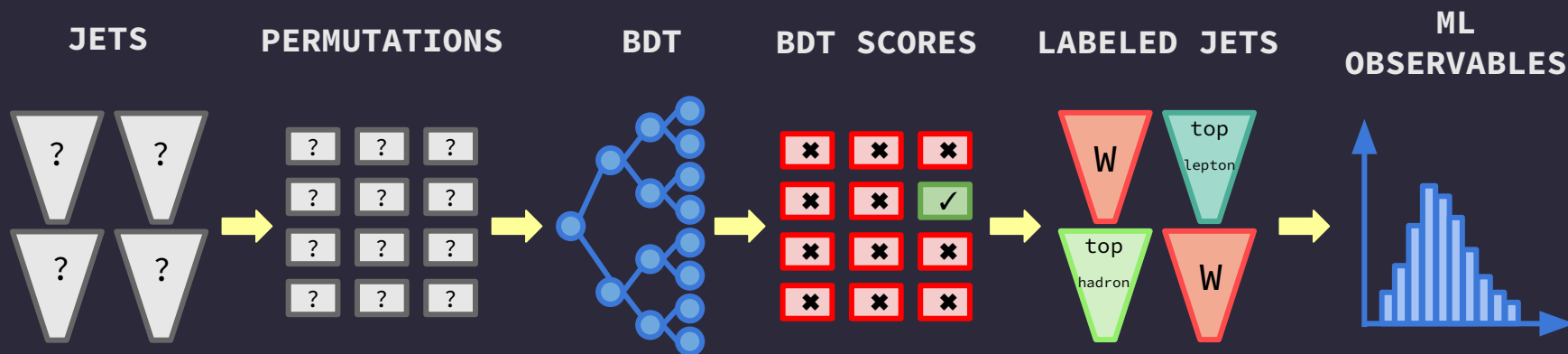
# Machine Learning Task

- Assign jets to their parent partons
- Allows us to approximate observables such as
  - ◆ Mass of top quark (combined mass of  $b_{\text{top-had}}$  and two  $W$  jets)
  - ◆ Angle between  $\text{top}_{\text{lepton}}$  jet and lepton ( $\Delta\phi$ )



# Approach to ML Task

1. Consider **leading N jets** in each event
2. Find **all possible permutations** of parton assignments of these N jets (two W, one  $b_{\text{top-had}}$ , one  $b_{\text{top-lep}}$ )
3. **Calculate features** for each set of permutations and **feed into BDT**
4. Select permutation with **highest BDT score**
5. Use selection to **label jets**
6. Calculate **ML observables**







# Training





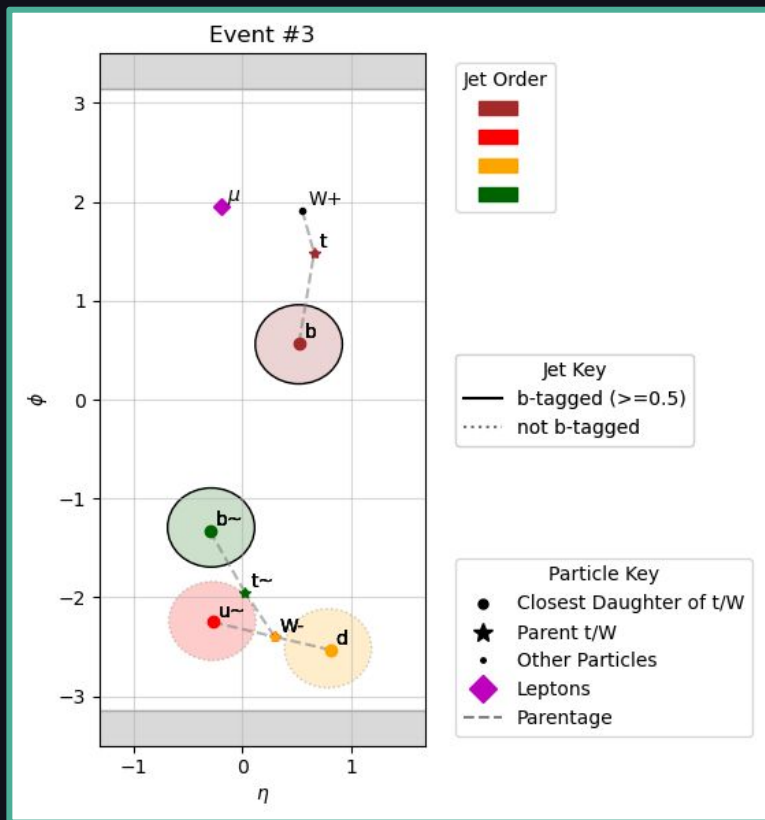
# Training: Event Selection



- Only look at ttbar MC events
- Match jets to particles using  $\Delta R$  matching
- Find parent particle (either W or top quark)
- Keep events which are 100% reconstructable

```
nearest_genpart = jets.nearest(genparts, threshold=0.4)
nearest_parent = nearest_genpart.distinctParent # parent of matched particle
parent_pdgid = nearest_parent.pdgId # pdgId of parent particle
```

Example Event





# Training: Obtaining Training Features

Python HEP libraries utilized:

- [uproot](#)
- [coffea](#)
- [awkward](#)

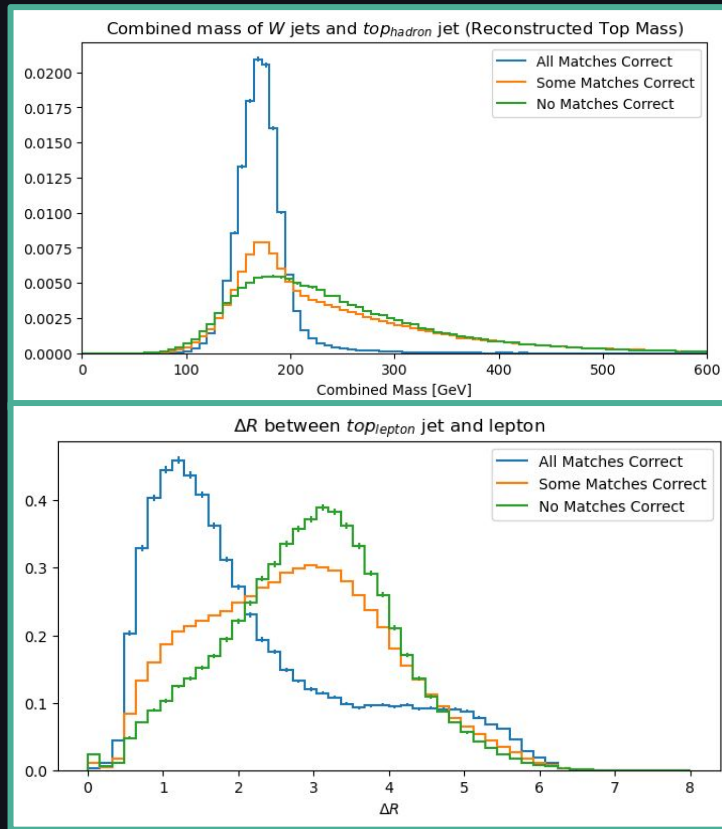
coffea processor with  
`coffea.nanoevents.NanoAODSchema`

use `awkward` to calculate  
features



`coffea.processor.column_accumulator`  
objects

EXAMPLE FEATURES





# Training: Hyperparameter Optimization



Set up `mlflow` on `dask` workers

Map jobs to workers (each job trains models for each hyperparameter selection)

Number of times to split dataset for cross-validation

```
client = utils.get_client()
client.run(initialize_mlflow)

futures = client.map(fit_model,
                    samples_even,
                    features=features_even,
                    labels=labels_even,
                    evaluation_matrix=evaluation_matrix,
                    n_folds=N_FOLD,
                    mlflowclient=mlflowclient,
                    use_mlflow=USE_MLFLOW,
                    log_models=MODEL_LOGGING)

res = client.gather(futures)
```

Track input hyperparameters

```
mlflowclient.log_param(run_id, param, np.average(value))
```

Track training scores

```
mlflowclient.log_metric(run_id, metric, np.average(value))
```

Log resulting model

```
with mlflow.start_run(run_id=run_id, nested=True) as run:
    mlflow.xgboost.log_model(result["model"], "model", signature=signature)
```

} For each trial



# Training: Hyperparameter Optimization

mlflow 2.2.1 Experiments Models GitHub Docs

**optimize-reconstruction-bdt-00** Provide Feedback Share

Experiment ID: 2 Artifact Location: mlflow-artifacts/2

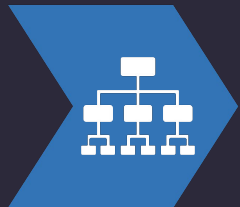
> Description Edit

Table view | Chart view |  | Sort: Created | Columns | Refresh

Time created: All time | State: Active

						Metrics			Parameters			
<input type="checkbox"/>	<input type="checkbox"/>	Run Name	Created	Duration	Models	test_jet_score	test_precision	test_recall	gamma	learning_rate	max_depth	n_estimators
<input type="checkbox"/>	<input type="checkbox"/>	run-99	8 minutes ago	8.0min	xgboost	0.896	0.791	0.723	0.0784759...	0.8888888...	71	301
<input type="checkbox"/>	<input type="checkbox"/>	run-98	8 minutes ago	8.0min	xgboost	0.899	0.782	0.744	0.0127427...	0.8888888...	41	451
<input type="checkbox"/>	<input type="checkbox"/>	run-97	8 minutes ago	7.9min	xgboost	0.898	0.782	0.736	0.0001	0.6666666...	61	401
<input type="checkbox"/>	<input type="checkbox"/>	run-96	8 minutes ago	7.8min	xgboost	0.901	0.789	0.739	2.9763514...	0.5555555...	71	351
<input type="checkbox"/>	<input type="checkbox"/>	run-95	8 minutes ago	7.7min	xgboost	0.9	0.772	0.721	0.2636650...	0.3333333...	1	401
<input type="checkbox"/>	<input type="checkbox"/>	run-94	8 minutes ago	7.7min	xgboost	0.854	0.731	0.683	0.4832930...	0.2222222...	21	1
<input type="checkbox"/>	<input type="checkbox"/>	run-93	8 minutes ago	7.6min	xgboost	0.902	0.789	0.741	0.0037926...	0.6666666...	21	401
<input type="checkbox"/>	<input type="checkbox"/>	run-92	8 minutes ago	7.5min	xgboost	0.892	0.762	0.709	0.0069519...	1.0	1	51
<input type="checkbox"/>	<input type="checkbox"/>	run-91	8 minutes ago	7.5min	xgboost	0.893	0.778	0.689	0.0011288...	0.8888888...	1	151
<input type="checkbox"/>	<input type="checkbox"/>	run-90	8 minutes ago	7.4min	xgboost	0.908	0.804	0.754	0.0784759...	0.2222222...	21	251

100 matching runs



# Training: Model Storage

```
best_model_path = f'runs://{best_run_id}/model'  
best_model = mlflow.xgboost.load_model(best_model_path) # load model locally  
result = mlflow.register_model(best_model_path, "reconstruction-bdt") # register best model in mlflow model repository
```

mlflow 2.2.1 Experiments Models GitHub Docs

Registered Models >

## reconstruction-bdt

Created Time: 2023-04-19 11:08:41 Last Modified: 2023-04-19 11:12:42

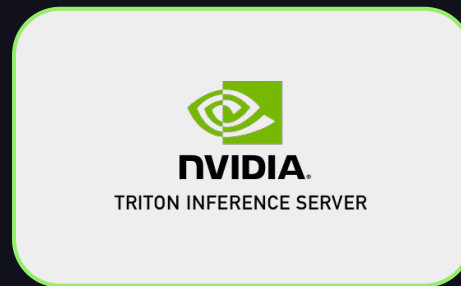
▼ Description Edit

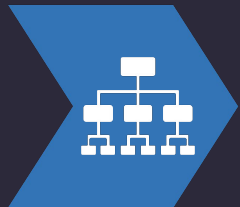
This is an xgboost boosted decision tree which classifies a permutation of jets within an event as "correct" or "incorrect". The permutation of jets corresponds to labelling the jets according to their parent partons (W, top\_hadron, or top\_lepton)

> Tags

▼ Versions All Active 2 Compare

<input type="checkbox"/>	Version	Registered at	Created by	Stage	Description
<input type="checkbox"/>	✔ Version 6	2023-04-19 11:10:20		Production	
<input type="checkbox"/>	✔ Version 5	2023-04-19 11:10:15		Production	
<input type="checkbox"/>	✔ Version 4	2023-04-19 11:10:11		Archived	





# Training: Model Storage



model\_name



```
name: "reconstruction_bdt_xgb"
backend: "fil"
max_batch_size: 50000000
input [
  {
    name: "input_0"
    data_type: TYPE_FP32
    dims: [ 20 ]
  }
]
output [
  {
    name: "output_0"
    data_type: TYPE_FP32
    dims: [ 2 ]
  }
]
instance_group [{ kind: KIND_GPU }]
parameters [
  {
    key: "model_type"
    value: { string_value: "xgboost" }
  },
  {
    key: "predict_proba"
    value: { string_value: "true" }
  },
]
version_policy: { all { }}
dynamic_batching { }
```



# Inference







# Inference: Machine Learning Inference

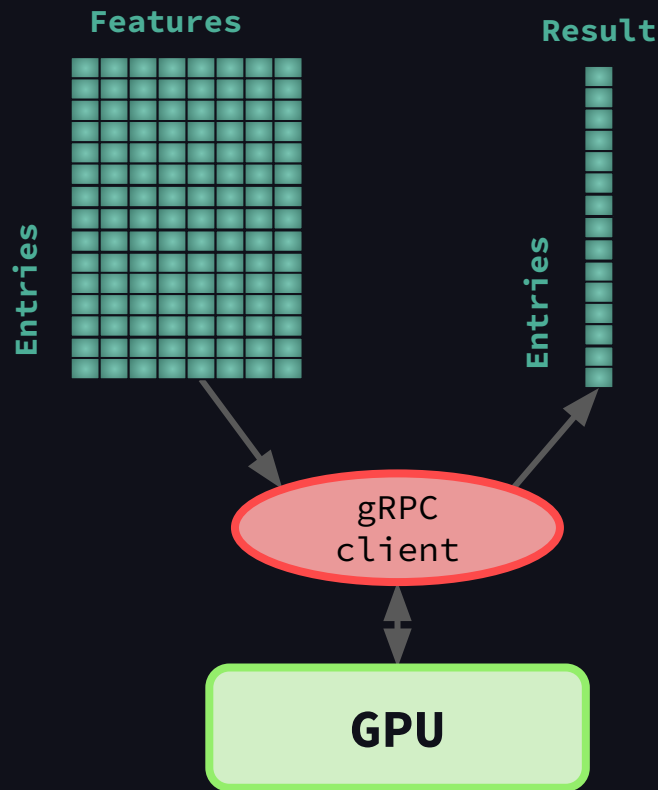


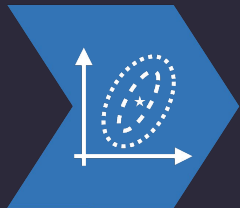
1. Set up `Triton` gRPC client
2. Perform inference request:

```
output = grpcclient.InferRequestedOutput(output_name)

inpt = [grpcclient.InferInput(input_name, features.shape, dtype)]
inpt[0].set_data_from_numpy(features)

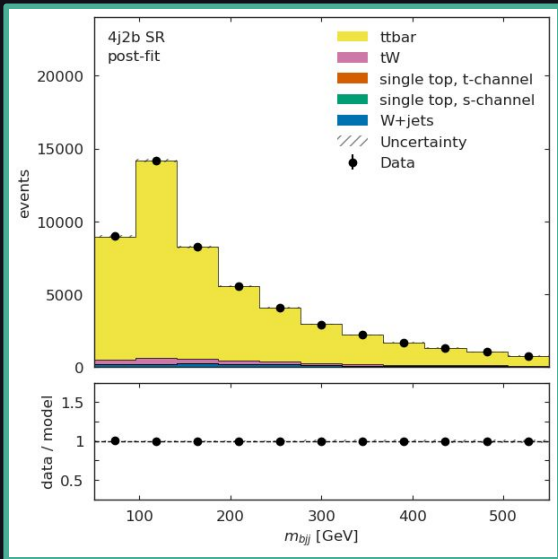
results[even_perm]=triton_client.infer(
    model_name=self.model_name,
    model_version=self.model_vers_odd,
    inputs=inpt,
    outputs=[output]
).as_numpy(output_name)[: , 1]
```



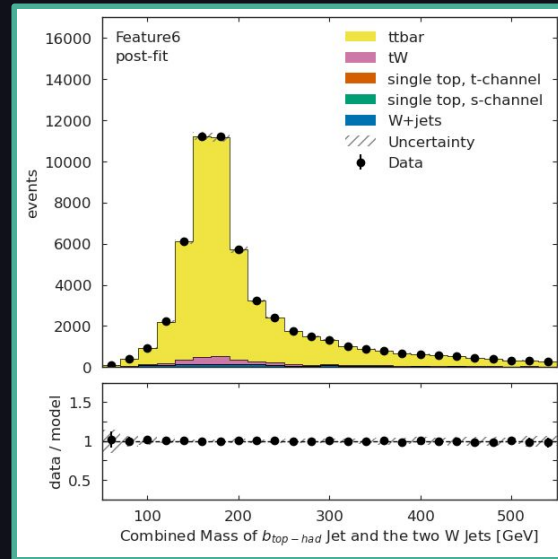


# Inference: Result, diagnostics

Statistical model fit from non-ML approach



Apply fit results to model with ML observables



Python HEP libraries utilized:

- [cabinetry](#)
- [hist](#)

```
model_prediction = cabinetry.model_utils.prediction(model_ml)
fit_results_mod = cabinetry.model_utils.match_fit_results(model_ml, fit_results)
model_prediction_postfit = cabinetry.model_utils.prediction(model_ml, fit_results=fit_results_mod)
```

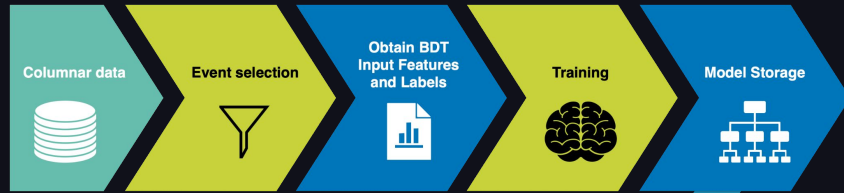


# Future Goals

→ Explore more complex ML models

- ◆ Now that we have the infrastructure in place, we can extend to deep learning approaches
- ◆ Triton inference server will help a lot here!

→ Benchmarking





[AGC Documentation](#)

Try it yourself!

[Coffea-casa @ UNL](#)

[Elastic Analysis  
Facility @ Fermilab](#)

[Local Setup](#)

Learn  
more  
about  
AGC!