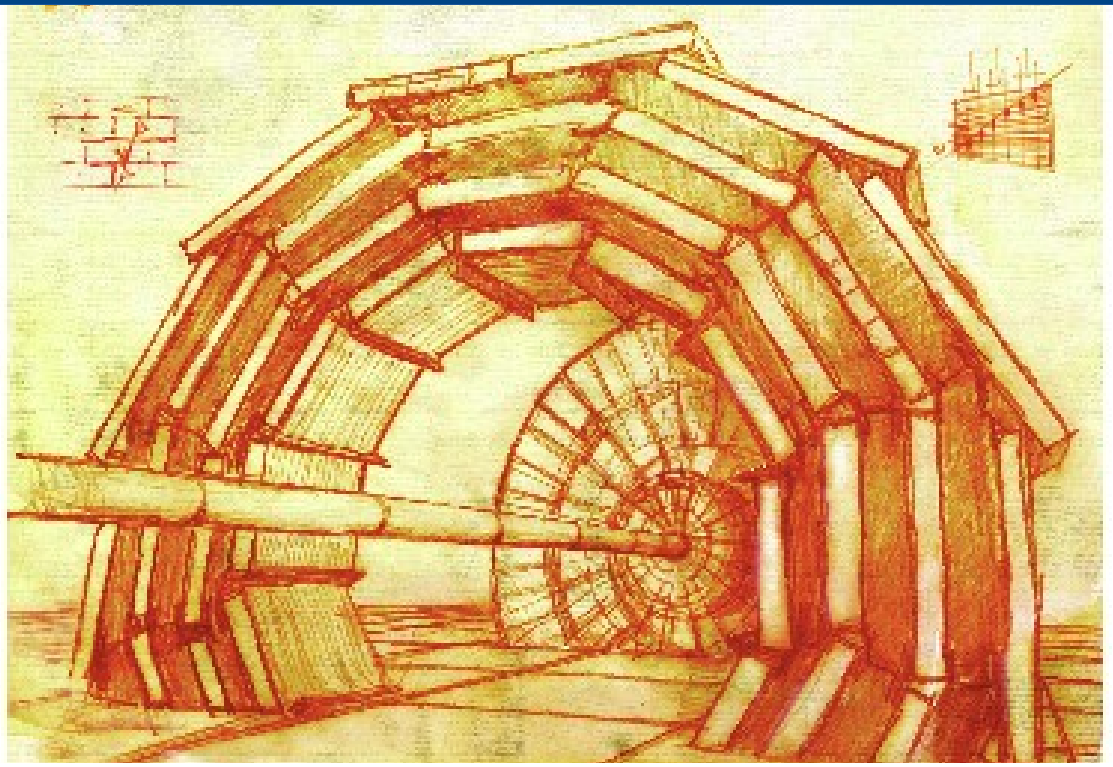
State of the ML Short Exercise



Matthias Komm, Jorn Bach

Reminder: Concept

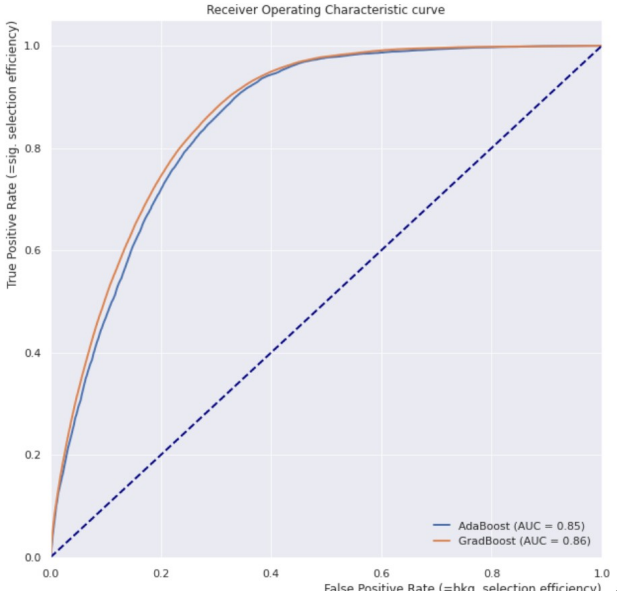
1) Exploratory Data Analysis, Classification with a BDT
2) DNNs and binary Classification
3) DNNs and multiclass Classification (optional 1)
4) DNNs and Regression (optional 2)
5) CNNs and Regression (optional 3)

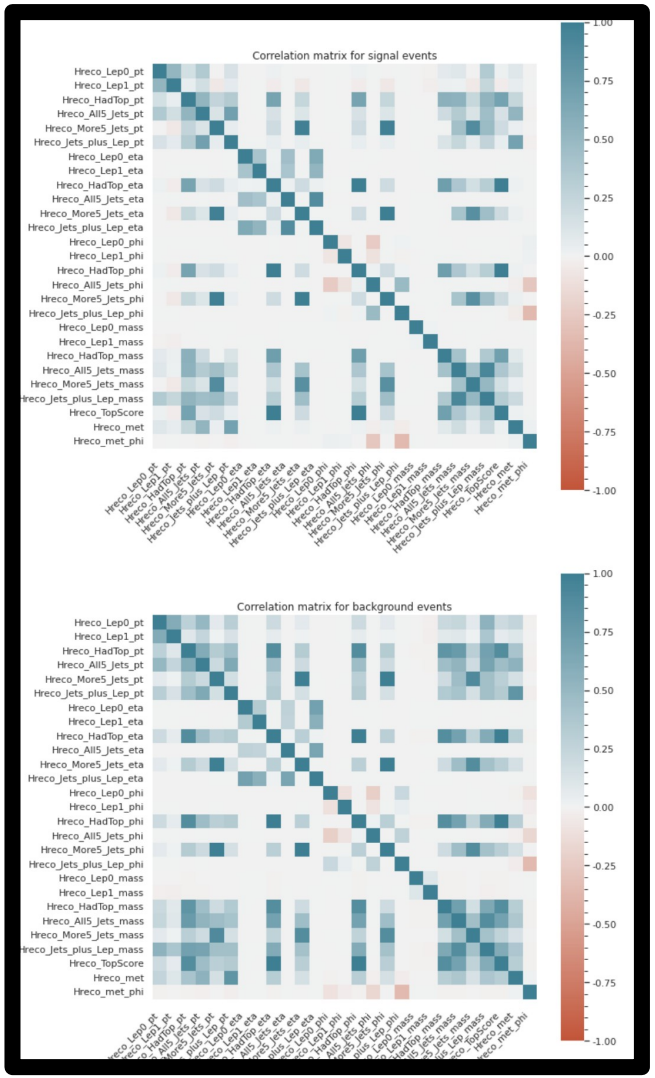
- Technical Details:
  - CERN SWAN Infrastructure with GPU nodes
  - Right now: Code and data on my EOS → get that to gitlab
  - jupyter notebooks

1. Exploratory Data Analysis

- Some snapshots from the exercise

```
plot_rocs([ [fitted_bdt_ada.decision_function(X_test), 'AdaBoost'],
            [fitted_bdt_grad.decision_function(X_test), 'GradBoost']],
          y_test)
```





Exercise : Data Preprocessing

We have worked on the raw data so far, yet we mentioned data preprocessing in an earlier cell in this notebook. Try your hand at some preprocessing using the sklearn library! Use either one of the preprocessing methods suggested below via code like `std_scaled_X_train = StandardScaler().fit_transform(X)` and retrain a BDT and compare its performance.

Another preprocessing option that you might want to try is the Principal Component Analysis.

Compare your results :)

```
[0]: from sklearn.preprocessing import (
        MaxAbsScaler, # maxAbs
        MinMaxScaler, # MinMax
        Normalizer, # Normalization (equal integral)
        StandardScaler# standard scaling
    )
    from sklearn.decomposition import PCA
```

Binary Classification with a DNN

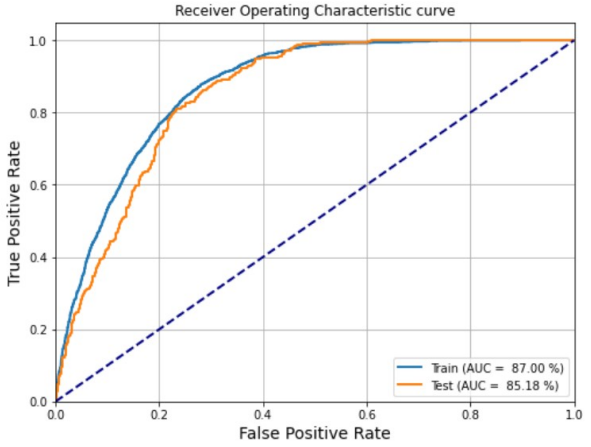
- Some exercise snapshots

```
[25]: epochs=100
learningRate = 0.01
optimizer = torch.optim.SGD(model.parameters(), lr=learningRate)
scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.9)

train_losses=[]
test_losses=[]
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train_loss=train_loop(train_dataloader, model, loss_fn, optimizer, scheduler, device)
    test_loss=test_loop(test_dataloader, model, loss_fn, device)
    train_losses.append(train_loss)
    test_losses.append(test_loss)
    print("Avg train loss", train_loss, ", Avg test loss", test_loss, "Current learning rate", scheduler.get_last_lr())
print("Done!")
```

```
Epoch 1
-------------------------------
100%|██████████| 20/20 [00:00<00:00, 155.45it/s]
100%|██████████| 4/4 [00:00<00:00, 230.07it/s]
Avg train loss 0.36067757 , Avg test loss 0.3366681635379791 Current learning rate [0.009000000000000001]
Epoch 2
-------------------------------
100%|██████████| 20/20 [00:00<00:00, 165.80it/s]
100%|██████████| 4/4 [00:00<00:00, 228.67it/s]
Avg train loss 0.3243506 , Avg test loss 0.3344101682305336 Current learning rate [0.008100000000000001]
Epoch 3
-------------------------------
100%|██████████| 20/20 [00:00<00:00, 168.16it/s]
100%|██████████| 4/4 [00:00<00:00, 237.74it/s]
Avg train loss 0.3202144 , Avg test loss 0.32744311541318893 Current learning rate [0.007290000000000001]
Epoch 4
-------------------------------
100%|██████████| 20/20 [00:00<00:00, 169.20it/s]
100%|██████████| 4/4 [00:00<00:00, 238.38it/s]
Avg train loss 0.3156821 , Avg test loss 0.3261355683207512 Current learning rate [0.006561000000000002]
```

```
plot_rocs([
    (model(torch.tensor(X_train.to_numpy(),device=model.device)).numpy(force=True), y_train, "Train"),
    (model(torch.tensor(X_test.to_numpy(),device=model.device)).numpy(force=True), y_test, "Test")
])
```



Status Conclusion

- Exercise notebooks are finished and work well
- Maybe we'll add an "avenues to explore" notebook but content amount is adequate already

- CERN SWAN should be a reliable option, also for GPU access
- students will need CERN account and 2FA
- We'll also include an option to run locally, modern laptops should be quick enough for this to be viable