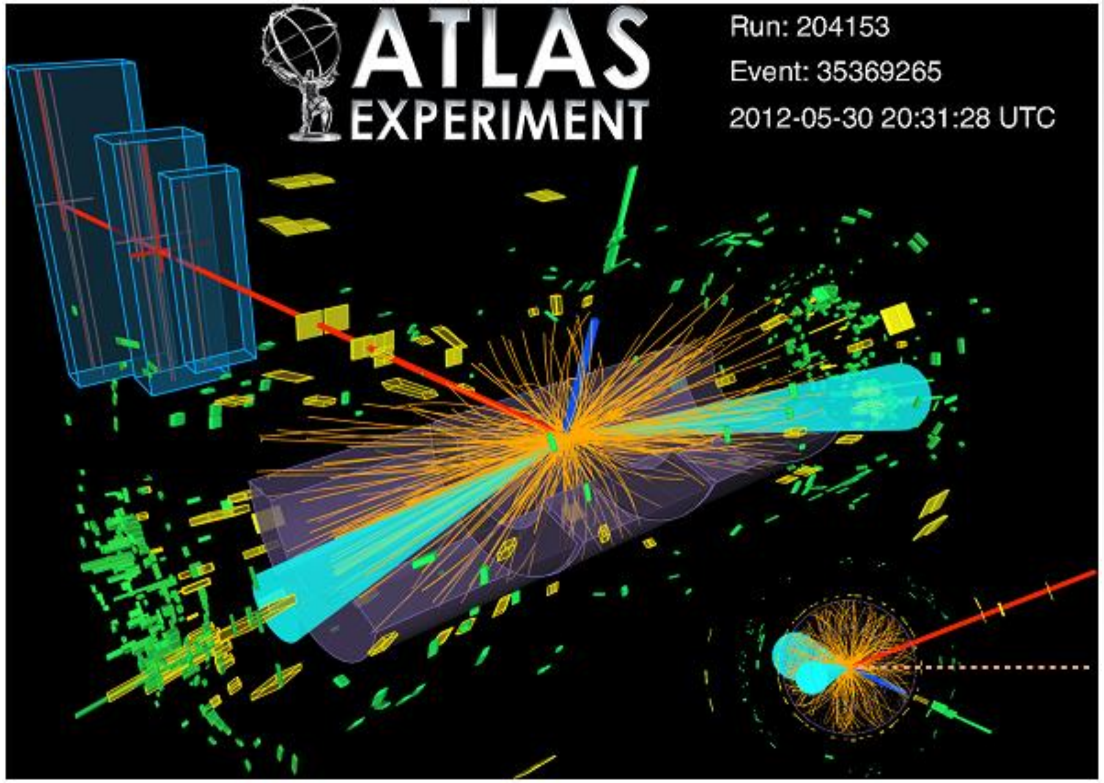
Chris Leon

Using Machine Learning to Identify the Higgs Boson

Higgs Boson Challenge

$$H \rightarrow \tau^+\tau^- \rightarrow e^\pm/\mu^\pm + 3\nu + \tau hadrons$$



- Simulated data created using Monte Carlo methods
- Kaggle competition 2014
- Goal to maximize:

$$AMS = \sqrt{2\left((s + b + b_r) ln\left(1 + \frac{s}{b+b_r}\right) - s\right)}$$

$$\approx s/\sqrt{b}$$

s,b are (weighted) TP and FP rate, $b_r = 10$

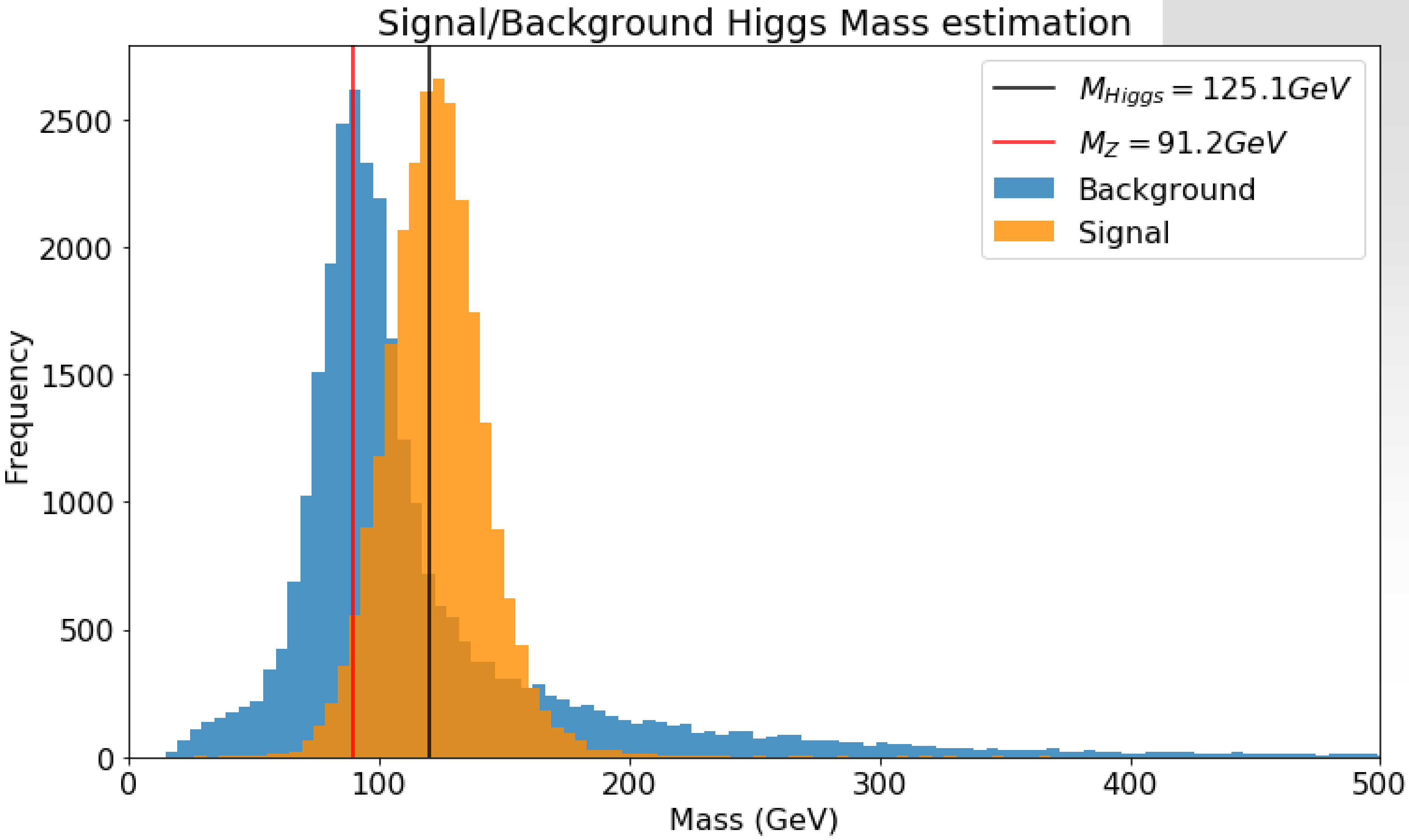
- Public: 250,000 events

  Private: 550,000 events

Higgs Boson Challenge

Problems:

- Other events can produce above products. E.g. $Z \rightarrow \tau^+\tau^-$
- Neutrinos not observed
- Missing data



30 predictor features

- Primitive variables. Mostly kinematics ($|\mathbf{p}|, \phi$ and η) for leptons, τ-hadrons. Also, number of jets.
- Derived variables. E.g., mass estimation of based on phase space integration

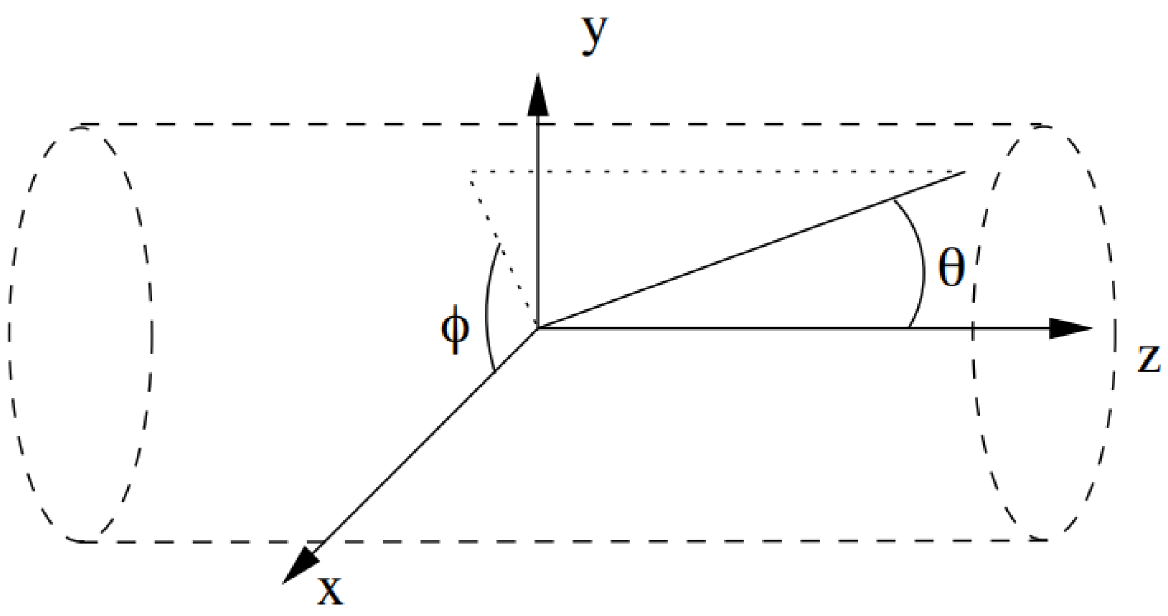
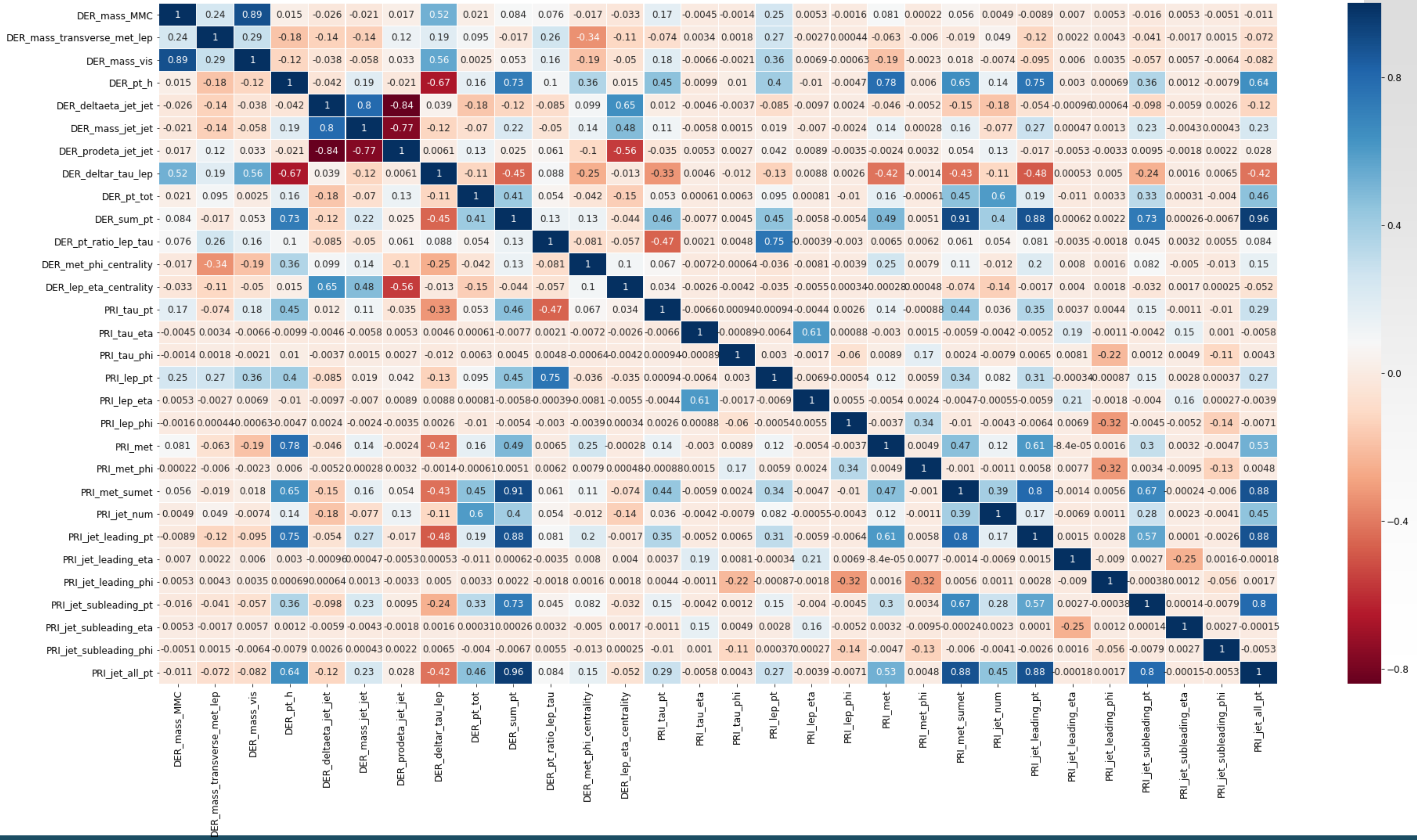


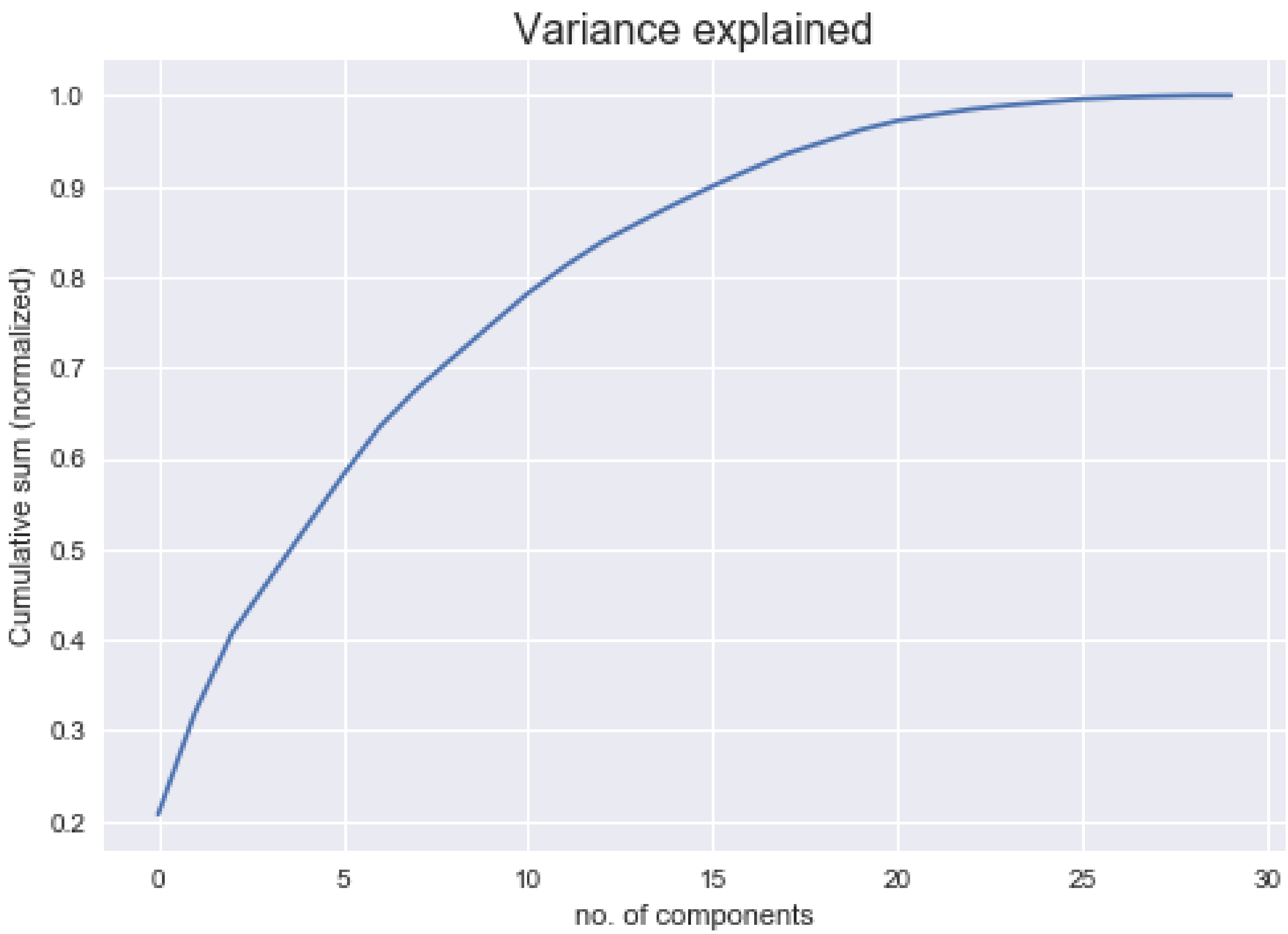
Figure 1: ATLAS reference frame

Data Exploration: Correlation Matrix



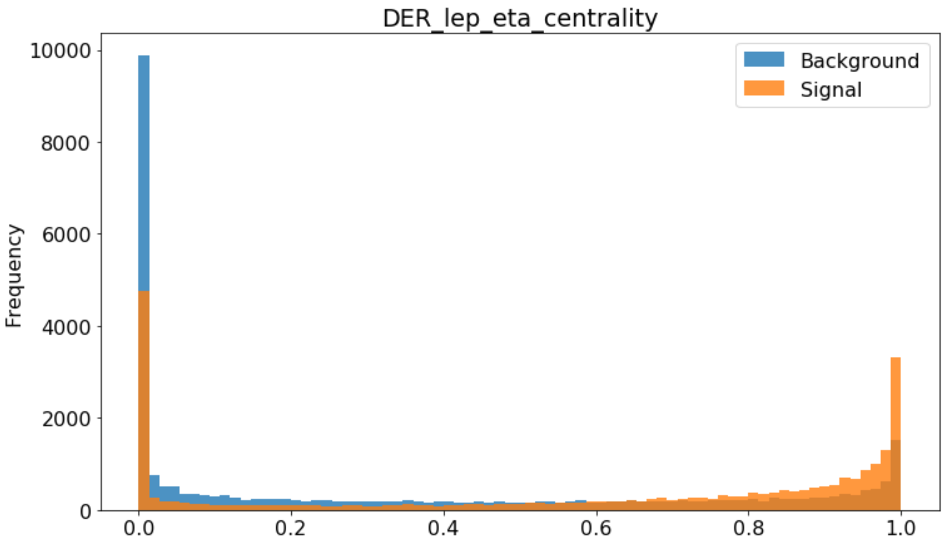
Principal Component Analysis

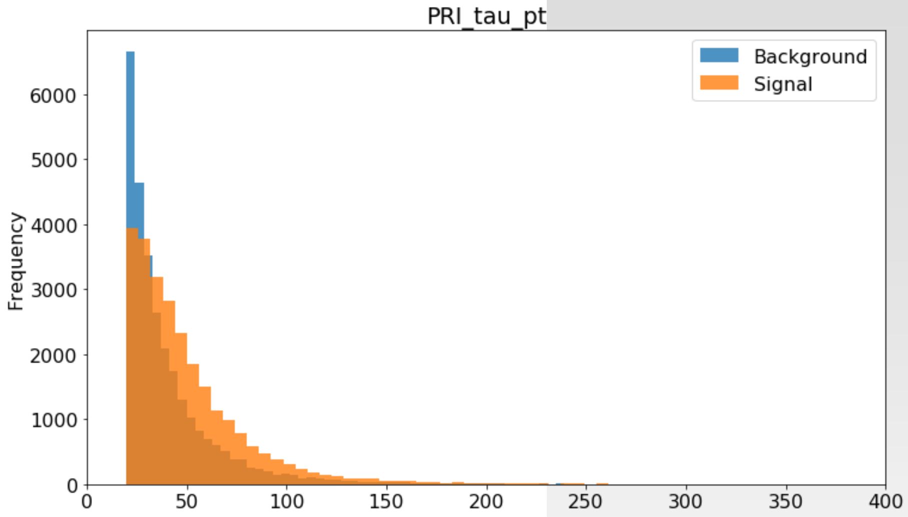
- PCA revealed that 25 components were driving nearly all the variance.



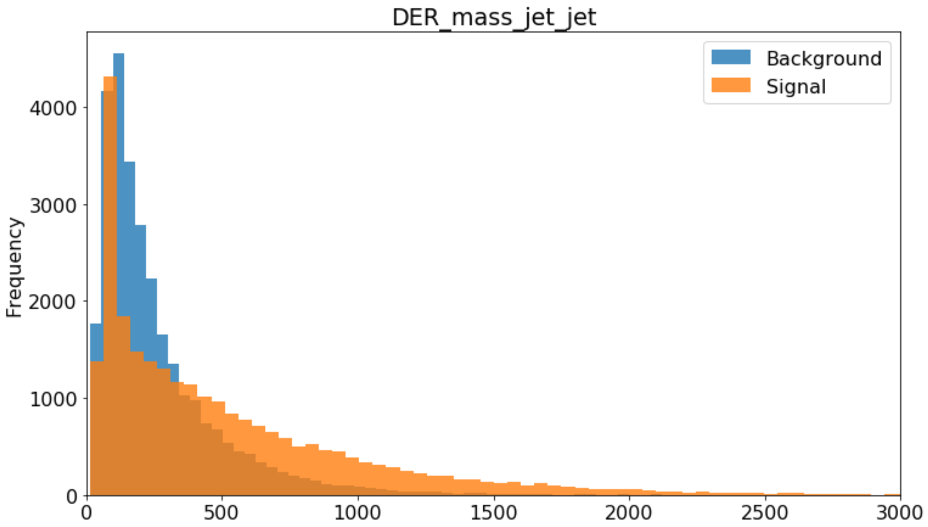
Data Exploration

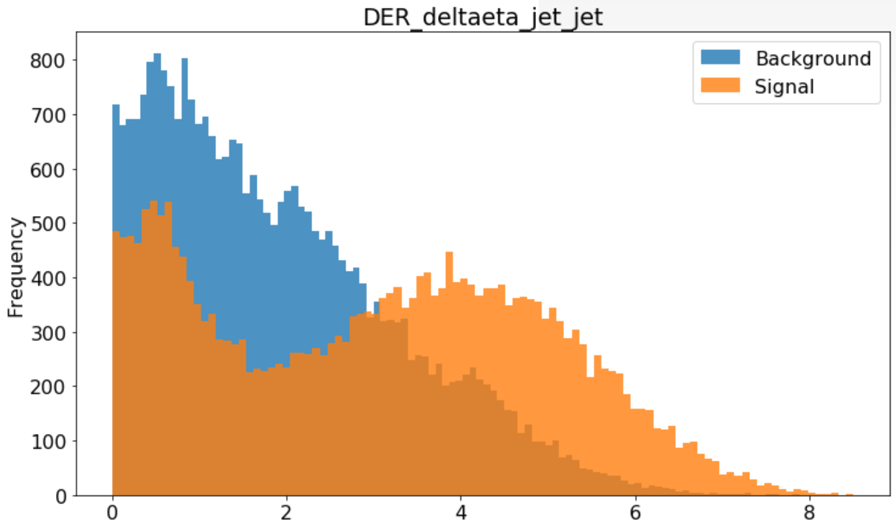
- On all features, means were all close relative to S.D.





- Signal and background had either similar distribution or heavy overlap





Drop 5 Azimuthal Angle ϕ Features

Theoretical Justification

- Problem has cylindrical symmetry

Empirical Justification

- Histograms consistent with uniform distribution for both signal and background
- Finally, ϕ 's correlated poorly with classification

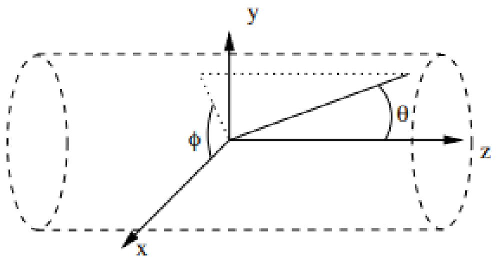
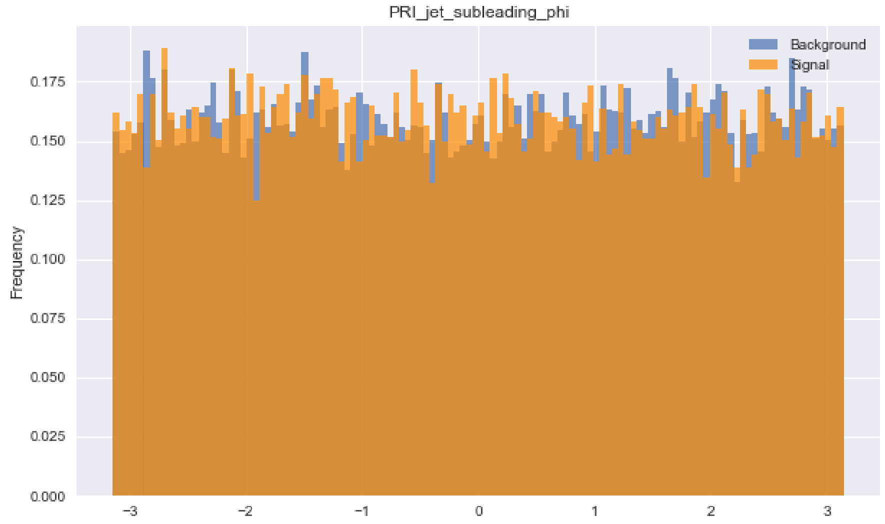
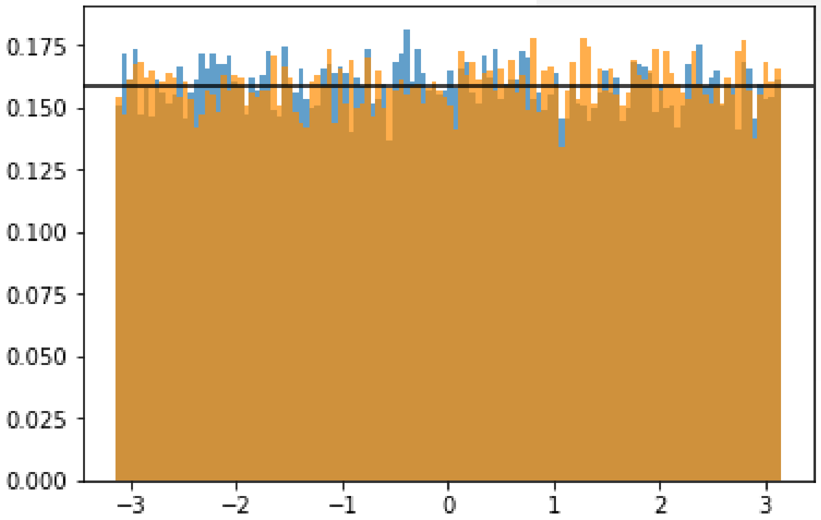


Figure 1: ATLAS reference frame

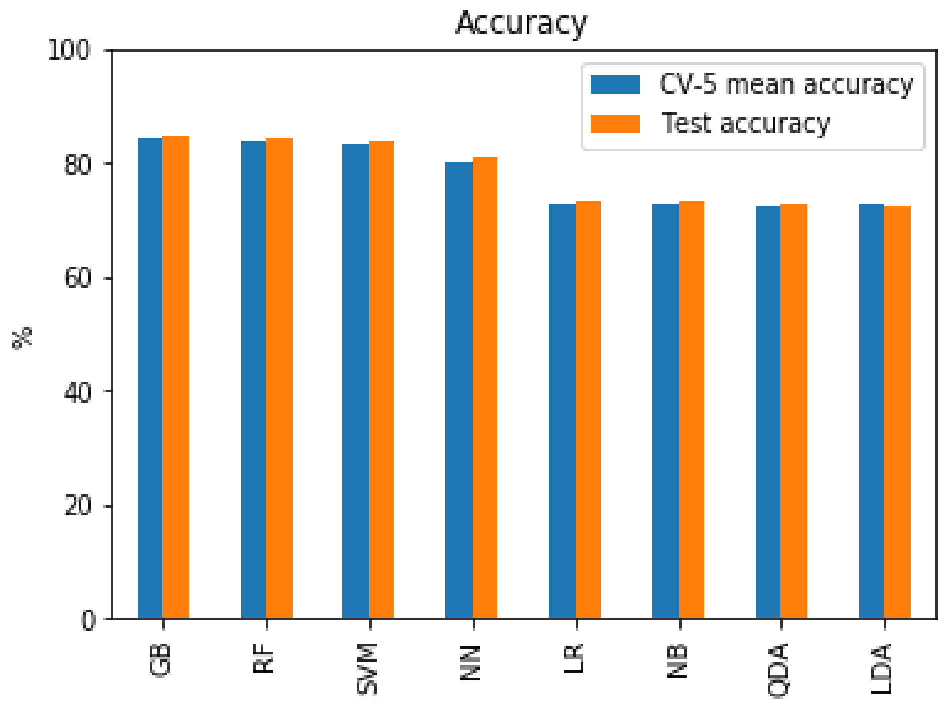




Methods

- Used scikit-learn library
- Tested several machine learning algorithms
- Divide data 80/20 into train/test.

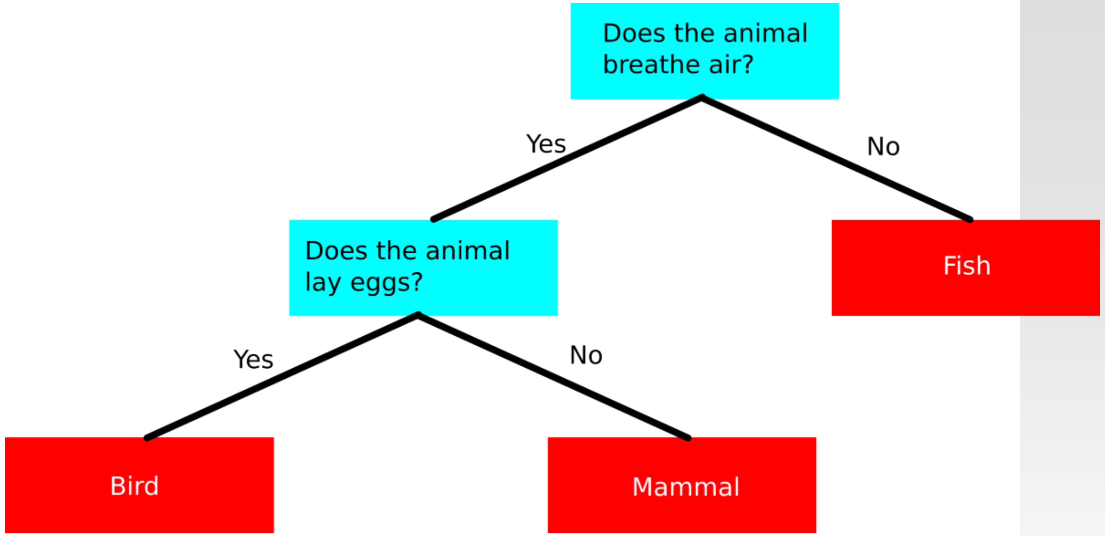
	CV-5 mean accuracy	Test accuracy
GB	84.384575	84.724363
RF	83.912960	84.540850
SVM	83.386246	83.814138
NN	80.099459	81.032078
LR	72.690904	73.515378
NB	72.692740	73.500697
QDA	72.246796	72.744623
LDA	72.689069	72.415628

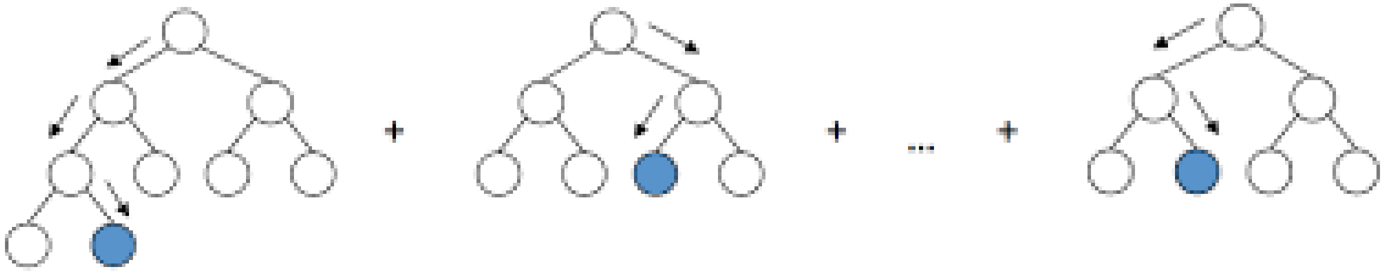


Example of Algorithm: Gradient Boosting

Gradient Boosted Trees

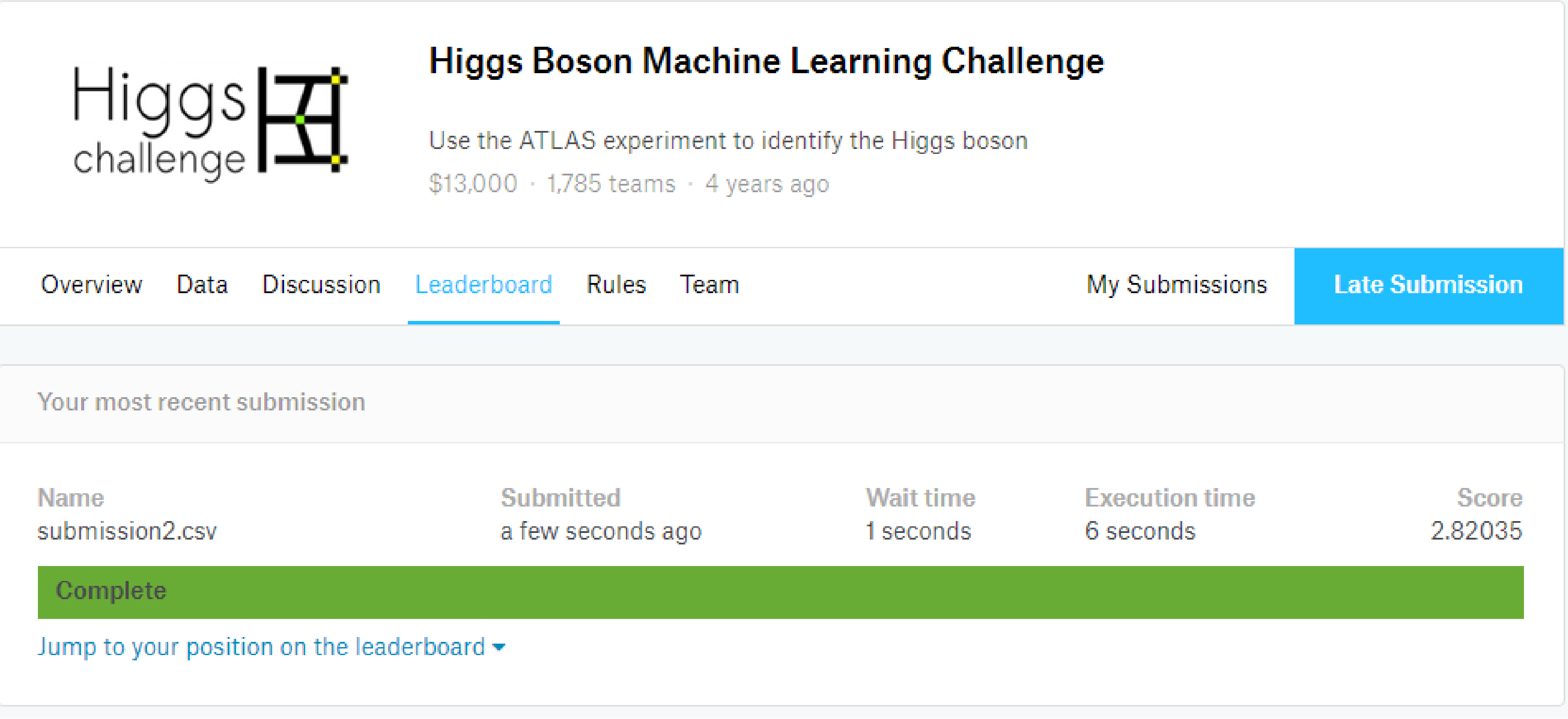
- Start with one decision tree (DT). Add one that tries to anticipate errors of first.
- Ensemble of DTs
- Hyperparameters: # of DTs, depth of trees, rate of learning, etc.





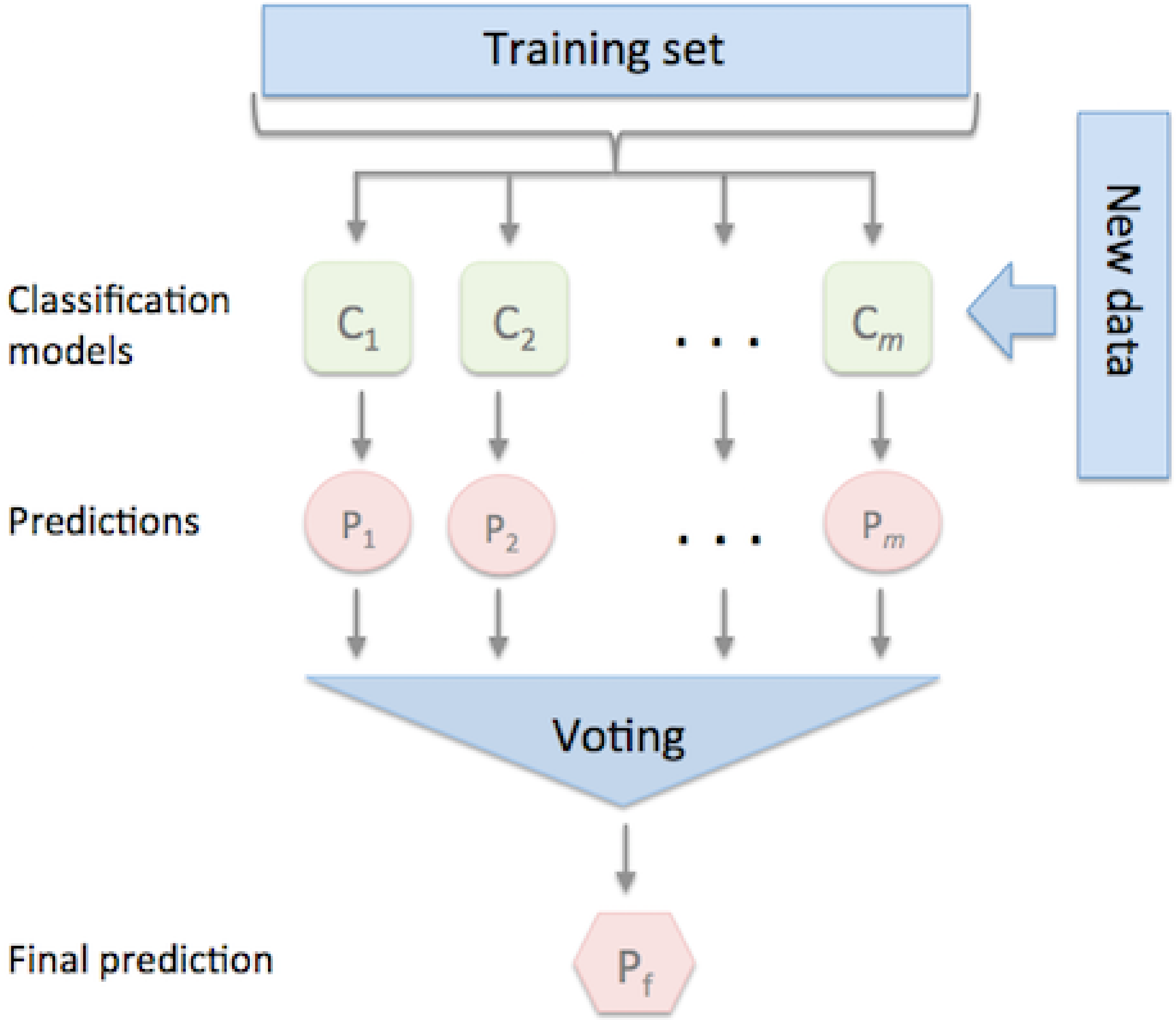
Higgs Challenge

- Trained on all 250,000 events.
- Filled missing values with means.
- Trained RF without ϕ's.
- AMS score: **2.82**
- LB: 1198 out of 1785



Ensemble voting

- Choose many models
- Each makes predictions
- Final prediction based on majority voting



Higgs Challenge

- Hyperparameter tuning
- Added other classifiers to voting ensemble
- Removed jet sub leading eta feature

- Up to AMS **3.03**, LB: 1021 out 1785

Voting Classifiers

Classifiers	Hyperparameters
RF	Max depth=12, n estimators=200
GB	Max depth = 13, n estimators=200
NN	2 Hidden layers: (100,10)
LR	C=1

Higgs Challenge

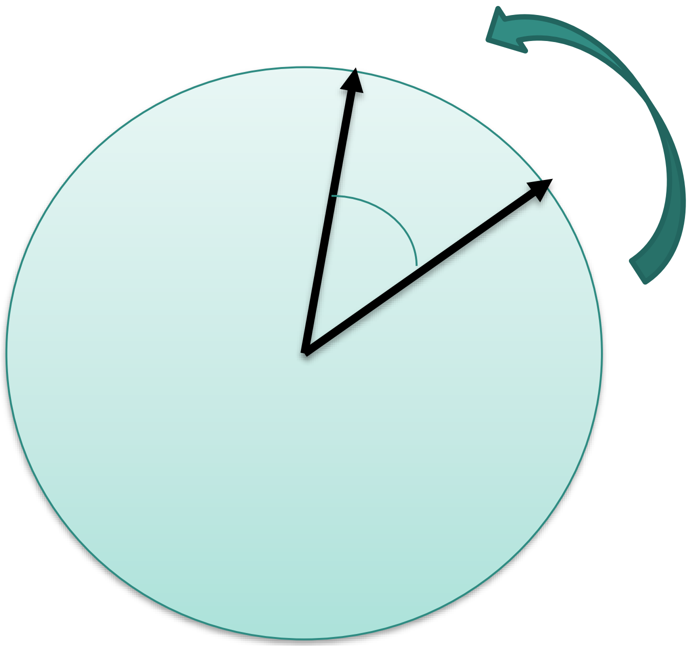
dmlc XGBoost

Read discussions. Users mentioned XGBoost, a library that:

- Optimized for GB
- Powerful and fast
- Deals automatically with missing values
- Can customize objective functions
- Had code that gets you AMS of 3.60.

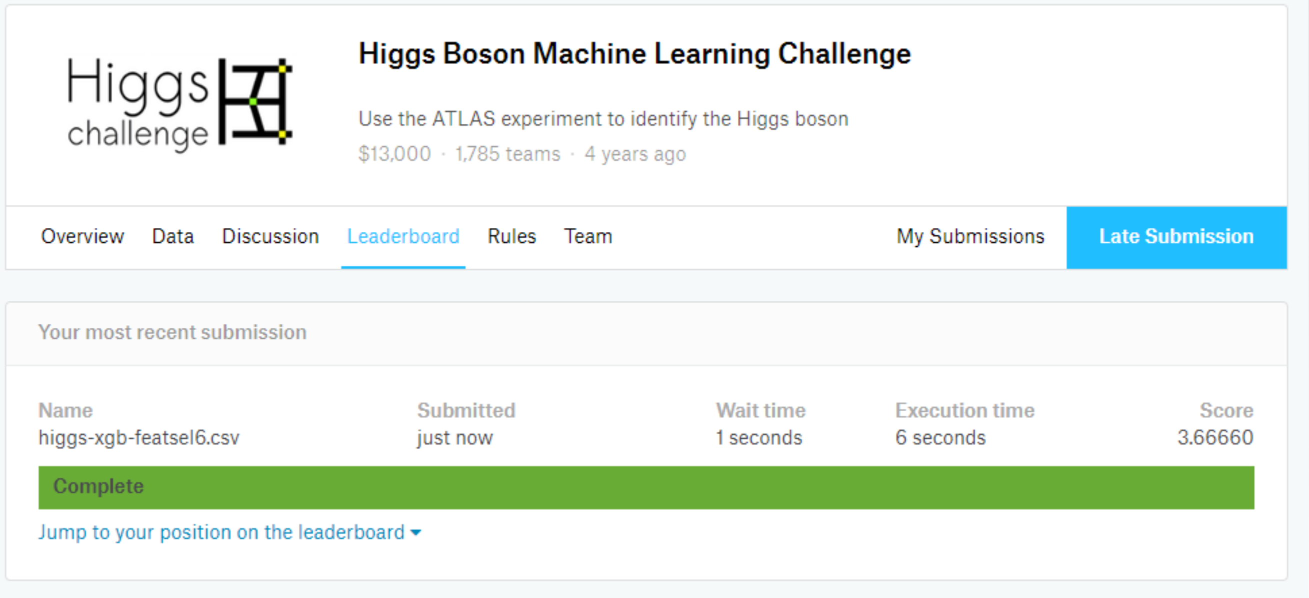
Also, users pointed out:

- Relative angles rotationally invariant



Higgs Challenge

- Looked at relative angles. Added some as features
- Got rid of ϕ's as before
- Did some hyperparameter tuning.



- AMS score: **3.67**

LB: 231 of 1785 (top 13%)

- Winner had AMS of 3.81

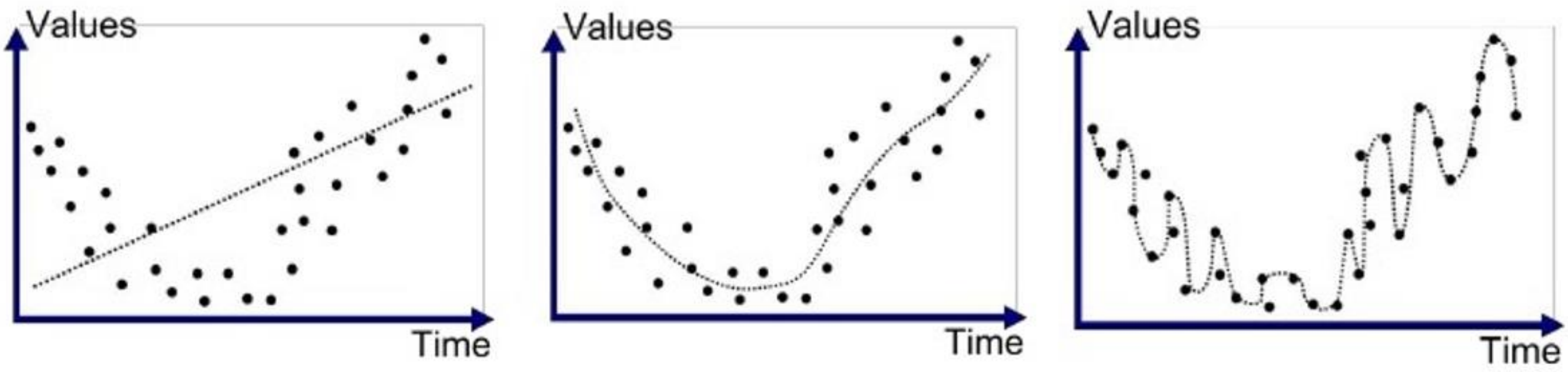
Conclusion

Takeaways:

- Feature selection and engineering are important and informed by physics
- Hyperparameter-tuning made big difference but expensive
- Consider tools/discussions
- If a theorist can do this, experimentalist should have no problem!

Issues (Backup slide)

Overfitting:



Underfitted | Good Fit/Robust | Overfitted

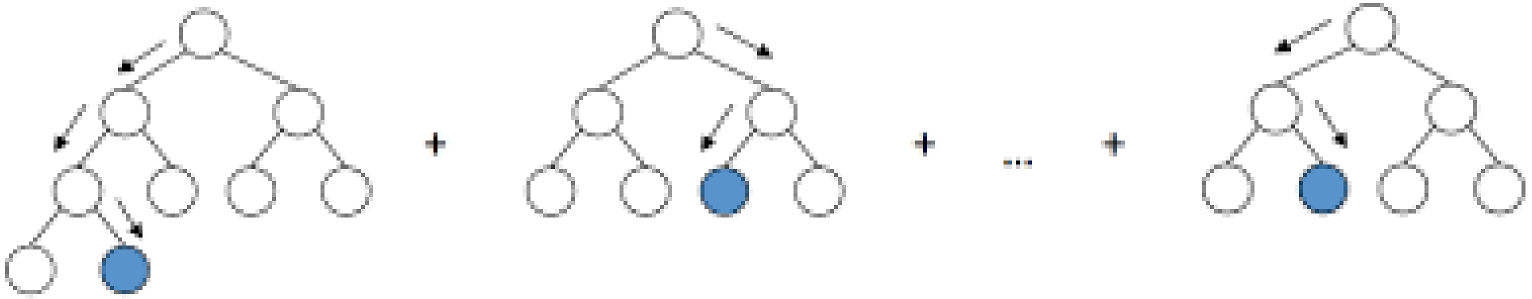
Running time (Backup slide)

- Best performers (RF, GB, SVM, NN) took much longer to fit to the data.
- Fitting using all data w/ mid-range laptop took 10 min. to 1 hour with RF,GB and NN. SVM took much longer.
- Hyper-parameter took very long and was computational expensive.
- *Computer made weird noises!* CPU at 100% use. ~ 1 hour w/ mid-high PC
- Only used 10k sample to hyperparameter tune.

Gradient Boosting (Backup slide)

Gradient Boosted Trees

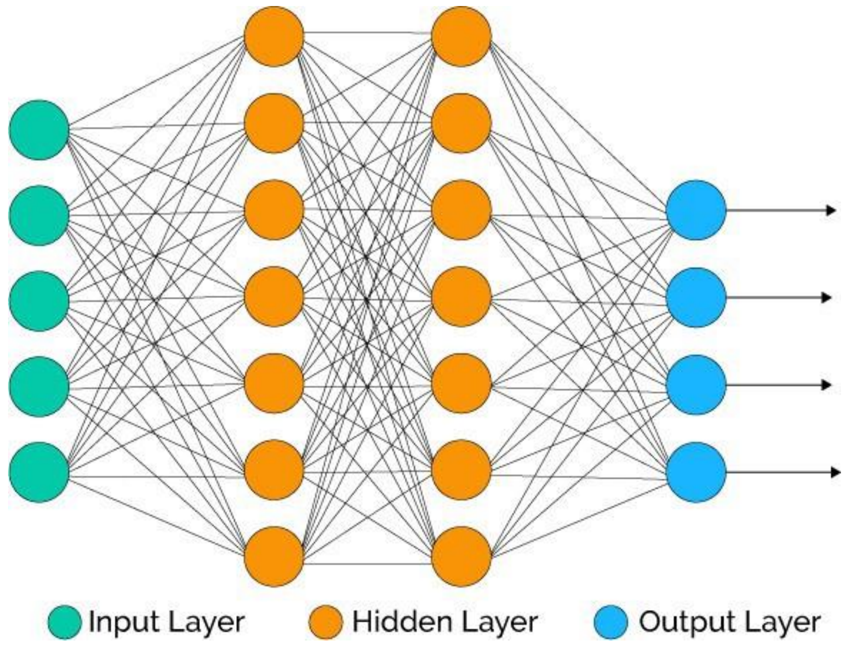
- Start with one decision tree (DT). Add one that tries to anticipate errors of first.
- 2 DTs decide by vote
- Add another that tries to anticipate error of first two, etc.



Other Algorithms (Backup slide)

❖ Neural networks

$$w_0 + \sum_i w_i x_i$$



❖ Random Forest