HPC for ML
Magurele Summer School

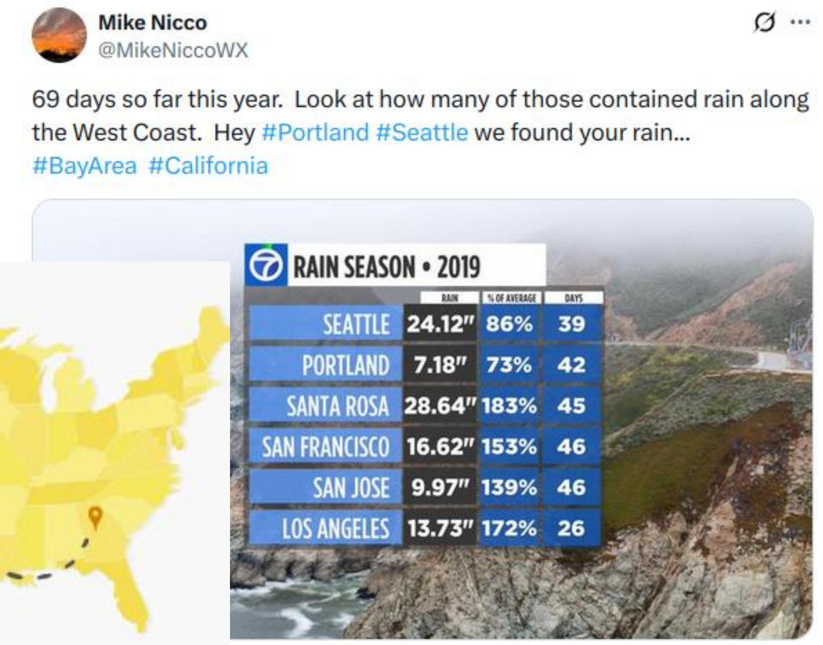
GPU Accelerated Distributed Data Science 3

Marco Celoria – CINECA

Magurele 9 July 2025

Exploratory data analysis (EDA)

- EDA is used for analyzing datasets and summarizing their main characteristics, using statistical graphics or visualization methods
- EDA is for seeing what the data can tell
- In the USA, and West Coast in particular, people argue about who has most rain
- What about the East Coast?
  We want to see if it rains more in Atlanta or Seattle



Mike Nicco
@MikeNiccoWX

69 days so far this year. Look at how many of those contained rain along the West Coast. Hey #Portland #Seattle we found your rain... #BayArea #California

RAIN SEASON • 2019

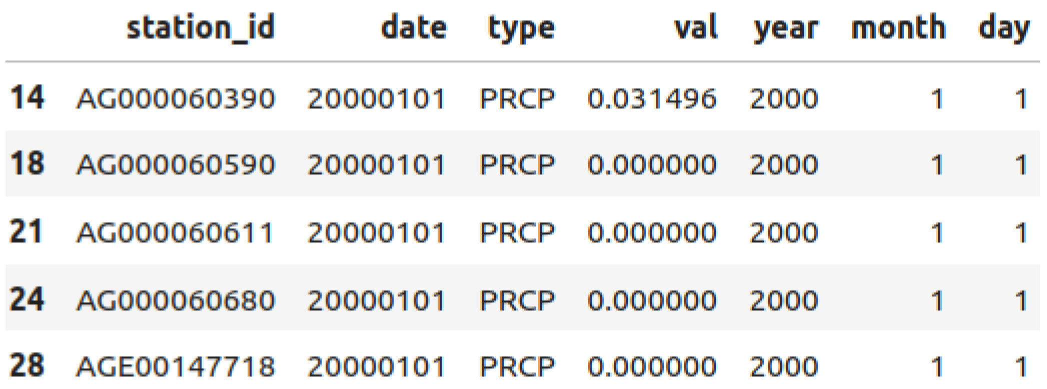
	RAIN	% OF AVERAGE	DAYS
SEATTLE	24.12"	86%	39
PORTLAND	7.18"	73%	42
SANTA ROSA	28.64"	183%	45
SAN FRANCISCO	16.62"	153%	46
SAN JOSE	9.97"	139%	46
LOS ANGELES	13.73"	172%	26

Exploratory data analysis (EDA)

- Data can be found at:
  https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt
  https://www.ncei.noaa.gov/pub/data/ghcn/daily/by_year/
- Load data and create a dataframe with the precipitation observations, but it's for all 100k weather stations, most of them nowhere near Atlanta, and this is time-series data, so we'll need to aggregate over time ranges.
- Then, we extract year, month, and day from the compound "date" column, so that we can compare total rainfall across time.
- Load up the station metadata file.
- There's no city in the station data, so we do some geo-math and keep only stations near Atlanta and Seattle
- Use a Groupby to compare changing precipitation patterns across time
- Use inner joins to filter the precipitation dataframe down to just Atlanta & Seattle data

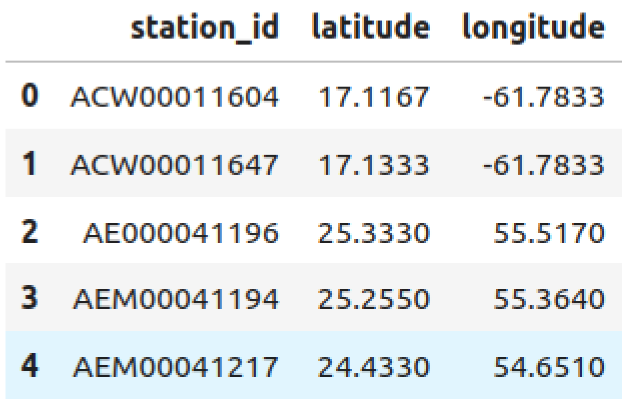
Exploratory data analysis (EDA)

- Extracting Finer Grained Date Fields



	station_id	date	type	val	year	month	day
14	AG000060390	20000101	PRCP	0.031496	2000	1	1
18	AG000060590	20000101	PRCP	0.000000	2000	1	1
21	AG000060611	20000101	PRCP	0.000000	2000	1	1
24	AG000060680	20000101	PRCP	0.000000	2000	1	1
28	AGE00147718	20000101	PRCP	0.000000	2000	1	1

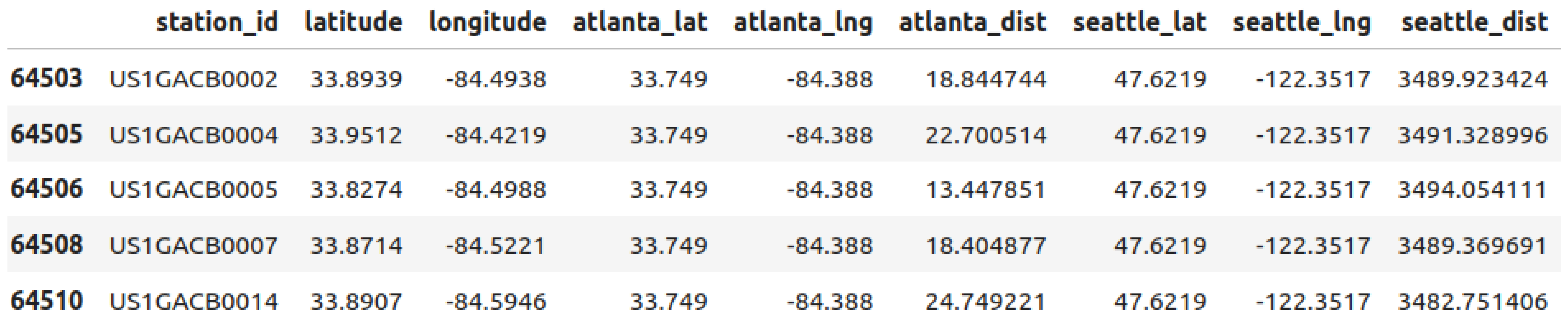
- Loading Station Metadata



	station_id	latitude	longitude
0	ACW00011604	17.1167	-61.7833
1	ACW00011647	17.1333	-61.7833
2	AE000041196	25.3330	55.5170
3	AEM00041194	25.2550	55.3640
4	AEM00041217	24.4330	54.6510

Exploratory data analysis (EDA)

- Filtering Weather Stations by Distance

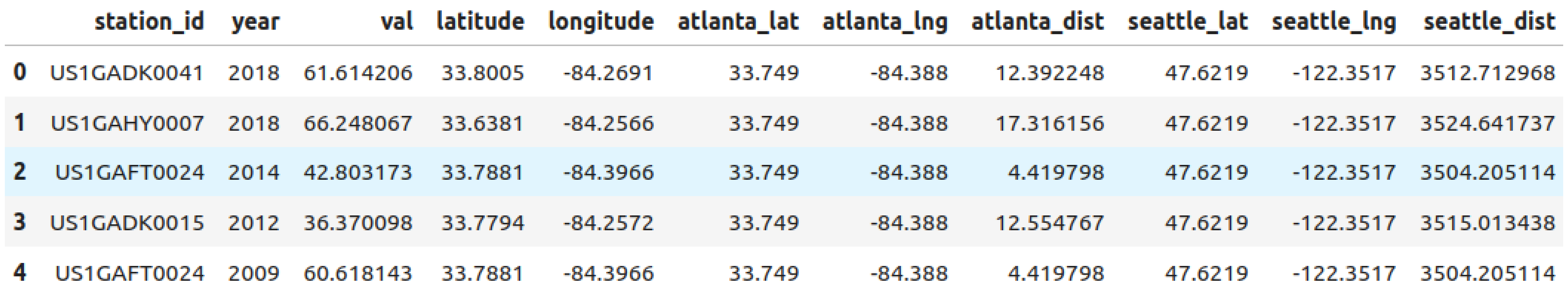


	station_id	latitude	longitude	atlanta_lat	atlanta_lng	atlanta_dist	seattle_lat	seattle_lng	seattle_dist
64503	US1GACB0002	33.8939	-84.4938	33.749	-84.388	18.844744	47.6219	-122.3517	3489.923424
64505	US1GACB0004	33.9512	-84.4219	33.749	-84.388	22.700514	47.6219	-122.3517	3491.328996
64506	US1GACB0005	33.8274	-84.4988	33.749	-84.388	13.447851	47.6219	-122.3517	3494.054111
64508	US1GACB0007	33.8714	-84.5221	33.749	-84.388	18.404877	47.6219	-122.3517	3489.369691
64510	US1GACB0014	33.8907	-84.5946	33.749	-84.388	24.749221	47.6219	-122.3517	3482.751406

- Grouping & Aggregating by Time Range
  Before using an inner join to filter down to city-specific precipitation data, we can use a groupby to sum the precipitation for station and year.

Exploratory data analysis (EDA)

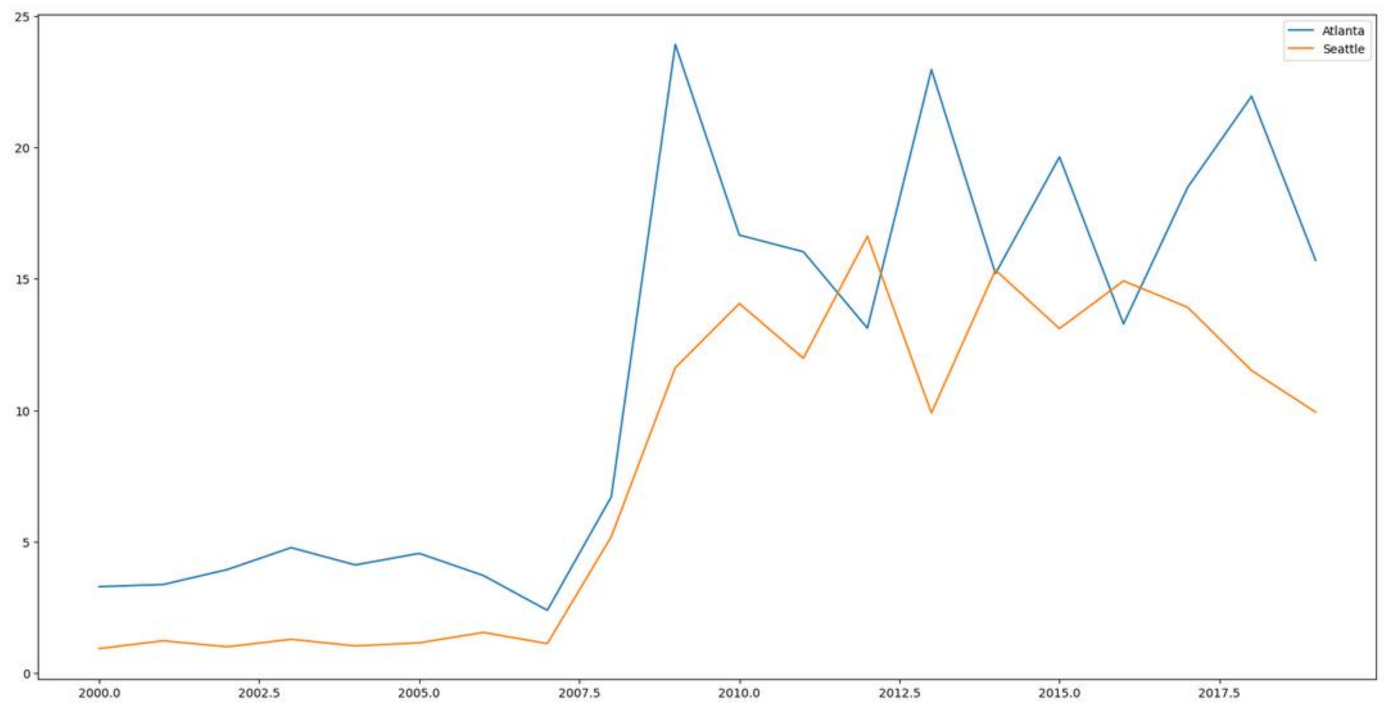
- Using Inner Joins to Filter Weather Observations
  We have separate DataFrames containing Atlanta and Seattle stations, and we have our total precipitation grouped by `station_id` and `year`. Computing inner joins can let us compute total precipitation by year for just Atlanta and Seattle.



	station_id	year	val	latitude	longitude	atlanta_lat	atlanta_lng	atlanta_dist	seattle_lat	seattle_lng	seattle_dist
0	US1GADK0041	2018	61.614206	33.8005	-84.2691	33.749	-84.388	12.392248	47.6219	-122.3517	3512.712968
1	US1GAHY0007	2018	66.248067	33.6381	-84.2566	33.749	-84.388	17.316156	47.6219	-122.3517	3524.641737
2	US1GAFT0024	2014	42.803173	33.7881	-84.3966	33.749	-84.388	4.419798	47.6219	-122.3517	3504.205114
3	US1GADK0015	2012	36.370098	33.7794	-84.2572	33.749	-84.388	12.554767	47.6219	-122.3517	3515.013438
4	US1GAFT0024	2009	60.618143	33.7881	-84.3966	33.749	-84.388	4.419798	47.6219	-122.3517	3504.205114

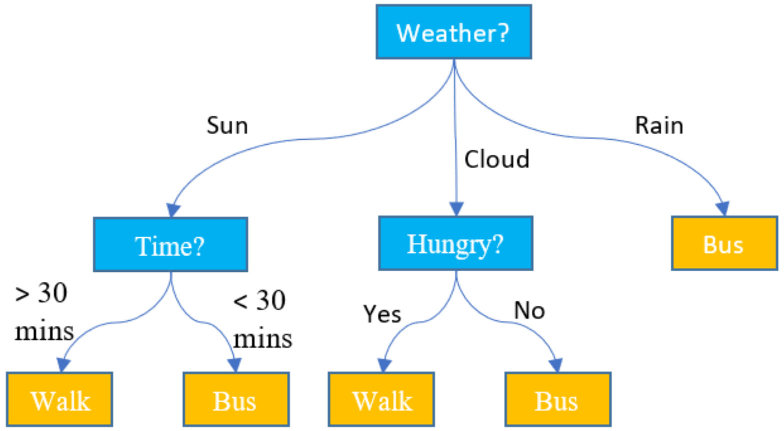
Exploratory data analysis (EDA)

- It looks like at least for roughly the last years, it rains more by volume in Atlanta than it does in Seattle



Decision trees

- A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks.
- It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.



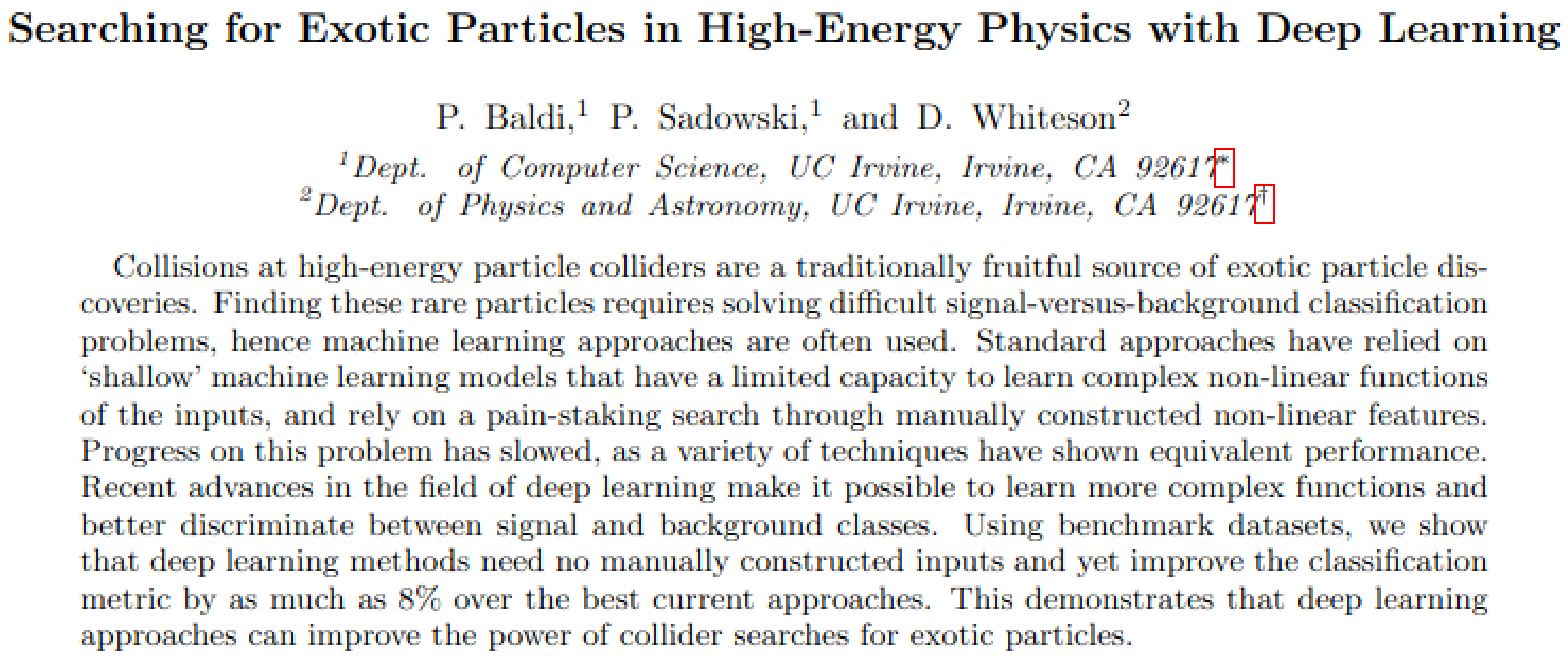
XGBoost and Random Forest

- XGBoost and Random Forest are both popular ensemble learning methods that build multiple decision trees to make predictions, but they differ in their approach and optimization.
- Random Forest uses a bagging technique, building trees independently and averaging their outputs
- XGBoost uses a boosting technique, building trees sequentially and correcting errors of previous trees.
- XGBoost is generally more accurate and efficient, especially with complex datasets, but requires more tuning and can be slower to train.

RAPIDS and XGBoost

- RAPIDS works closely with the XGBoost community to accelerateGradient Boosted Decision Trees on GPU on GPU
- XGBoost can load data from cuDF dataframes and cuPy arrays
- Dask allows XGBoost to scale to arbitrary numbers of GPUs

Searching for Exotic Particles in High-Energy Physics with Deep Learning



Searching for Exotic Particles in High-Energy Physics with Deep Learning

P. Baldi,[1] P. Sadowski,[1] and D. Whiteson[2]

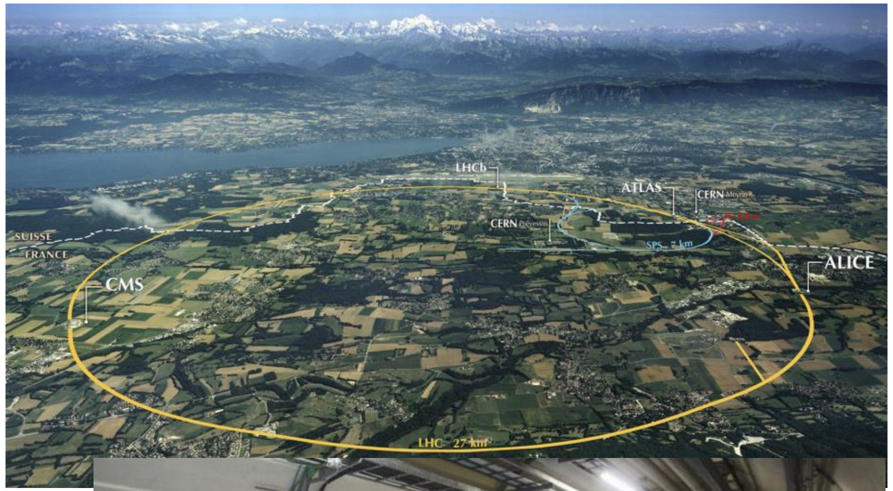
[1]*Dept. of Computer Science, UC Irvine, Irvine, CA 92617*[*]

[2]*Dept. of Physics and Astronomy, UC Irvine, Irvine, CA 92617*[†]

Collisions at high-energy particle colliders are a traditionally fruitful source of exotic particle discoveries. Finding these rare particles requires solving difficult signal-versus-background classification problems, hence machine learning approaches are often used. Standard approaches have relied on ‘shallow’ machine learning models that have a limited capacity to learn complex non-linear functions of the inputs, and rely on a pain-staking search through manually constructed non-linear features. Progress on this problem has slowed, as a variety of techniques have shown equivalent performance. Recent advances in the field of deep learning make it possible to learn more complex functions and better discriminate between signal and background classes. Using benchmark datasets, we show that deep learning methods need no manually constructed inputs and yet improve the classification metric by as much as 8% over the best current approaches. This demonstrates that deep learning approaches can improve the power of collider searches for exotic particles.

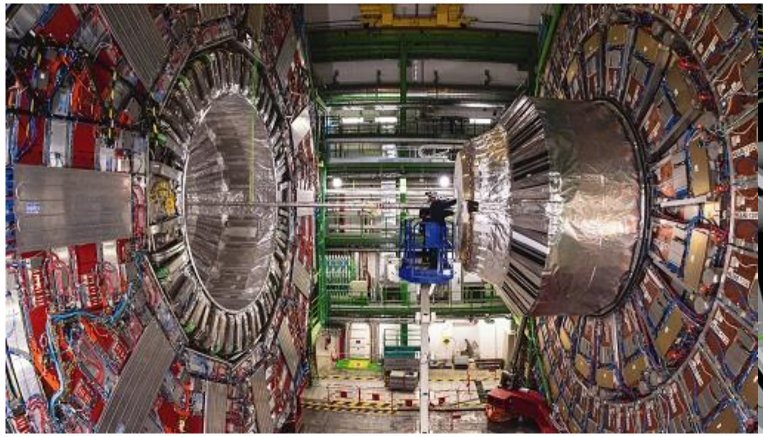
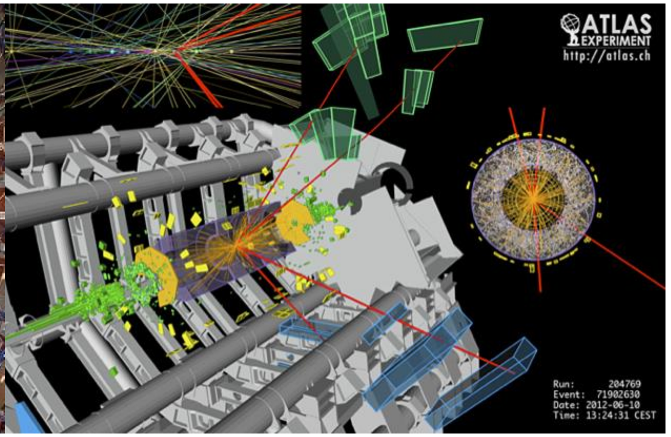
Higgs boson

- The field of high energy physics is devoted to the study of the elementary constituents of matter.
- The primary tools of high energy physicists are accelerators, which collide protons and/or antiprotons to create exotic particles.
- Discoveries require powerful statistical methods, and machine learning tools play a critical role.



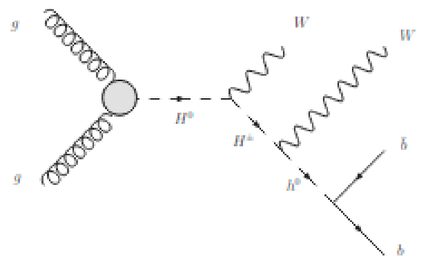
Higgs boson

- The majority of particle collisions do not produce exotic particles.
- Though the LHC produces approximately 10^{11} collisions per hour, approximately 300 of these collisions result in a Higgs boson.
- Good data analysis depends on distinguishing collisions which produce particles of interest (signal) from those producing other particles (background)

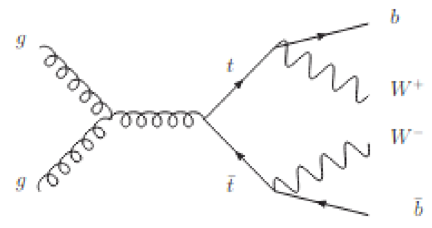



Higgs boson

- Our classification task is to distinguish between:
  - a signal process where new theoretical Higgs bosons are produced, and
  - a background process with the identical decay products but distinct kinematic features.
- Events are described by simple set of features which represent the measurements made by the detector: the momentum of each observed particle.
- In addition, we have missing transverse momentum in the event and b-tagging information for each jet
- These 21 features comprise our low-level feature set



(a)



(b)

Higgs boson

- Our knowledge of the different intermediate states of the two processes allows us to construct other features.
- As the difference in the two hypotheses lies mostly in the existence of new intermediate Higgs boson states, we can distinguish between the two hypotheses by attempting to identify whether the intermediate state existed.
- In addition to the low-level features, we consider 7 high-level features (functions of the 21 low-level features), derived by physicists (based on reconstructing the characteristic invariant mass) to help discriminate between the two classes.

Higgs boson

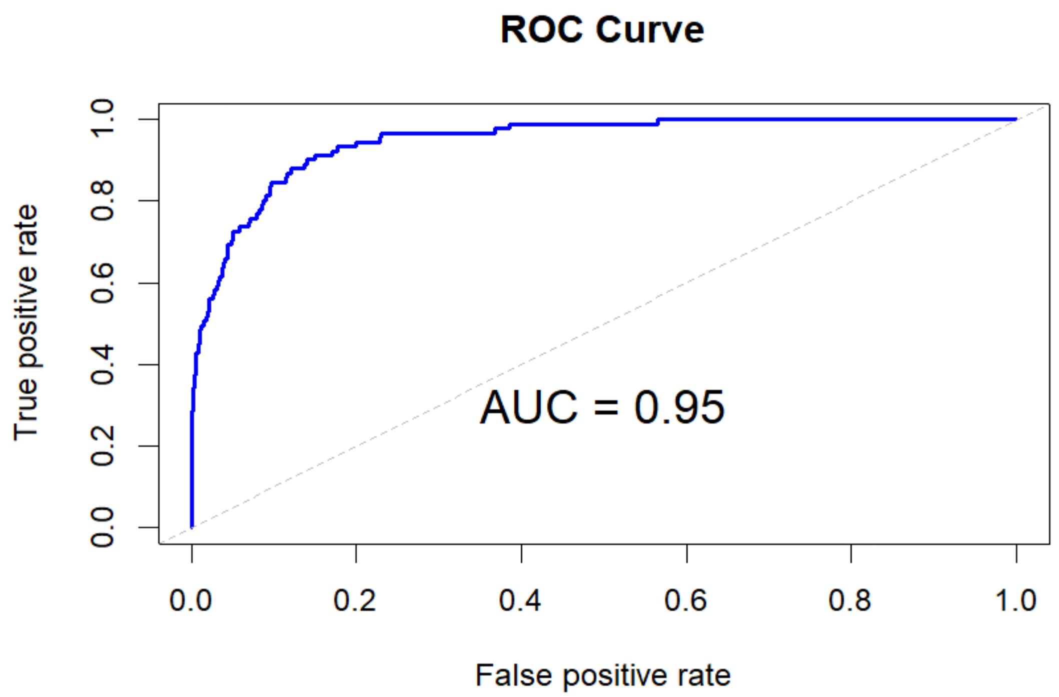
- The dataset can be found here: https://archive.ics.uci.edu/dataset/280/higgs
- We can build a distributed accelerated XGBoost classifier
- Without particular hyperparameter tuning, after training, we can get an accuracy of around 76% on the validation set and we compute the area under the ROC curve (AUC)
- The environment of high energy physics, with high volumes of relatively low-dimensional data containing rare signals hiding under enormous backgrounds, can inspire new developments in machine learning tools.

Threshold for binary classifier

- When we are predicting the probability of an instance belonging to a particular class, a threshold determines at what probability you classify an instance as the positive class (1) or the negative class (0).
- The Higgs classifier decides whether an event involves the Higgs boson or the background. The model computes a score (probability) for each event, representing the likelihood it's a Higgs.
- If we set ***a high threshold***, the model will only mark very obvious Higgs events, ensuring background events are not accidentally confused as Higgs events. This might mean some less-obvious Higgs events be classified as background.
- On the other hand, if you set a ***lower threshold***, the filter will catch a broader range of Higgs event, but there's also a higher chance it might incorrectly classify a background event as Higgs.

ROC

- A **ROC curve** is a graphical plot that illustrates the performance of a binary classifier model at varying threshold values.



- The ROC curve is the plot of the true positive rate (TPR) against the false positive rate (FPR) at each threshold setting.

AUC

- The Area Under the ROC Curve (AUC) is a metric used to evaluate the performance of a binary classification model.
- An AUC of 1.0 indicates a perfect model, while an AUC of 0.5 suggests the model performs no better than random guessing
  - **0.5:** No discriminatory power (random guessing).
  - **0.7 - 0.8:** Acceptable performance.
  - **0.8 - 0.9:** Excellent performance.
  - **0.9 - 1.0:** Outstanding performance.

New York taxi

- Consider the NYC Taxi & Limousine Commission yellow taxi data.
- The goal is to predict the fare amount for a given trip given the times and coordinates of the taxi trip using a Random Forest.

New York taxi

- Recall that on Leonardo, we need to download the dataset before starting the analysis
- The NYC Taxi & Limousine Commission yellow taxi is first loaded into a Dask Dataframe.
- The data is inspected, where it could be seen that the yellow taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, driver-reported passenger counts, distance, along with latitude, longitude, etc.
- These are the information that would be used to estimate the trip fare amount.

New York taxi

- The data needs to be cleaned up before it can be used in a meaningful way, in this part the user could recognize familiar pandas like functions such as (column) drop and fillna.
- New features are added like time difference between dropoff and pickup
- Just as with scikit-learn, the data could be easily split into training and test sets.
- Training data is then fitted to a Random Forest Model, and Inference is run on the test set.