Accelerated Machine Learning for Particle Discovery using NVIDIA RAPIDS Framework

*Abstract*— This research paper explores the practical implications and theoretical aspects of accelerated Machine Learning (ML) using the Python-based NVIDIA RAPIDS framework for particle discovery in large datasets. The paper covers the issues associated with such datasets, including data storage and machine learning algorithms used in other studies. It investigates parallel computing, distributed systems, and tools like Dask and RAPIDS, including CuDF and cuML. The paper provides a detailed data description, including EDA, data pre-processing, model selection, evaluation, and hyperparameter optimization. The paper concludes by evaluating the suitability of different processing architectures for specific computational tasks.

Keywords—RAPIDS, dask, parallel computing, distributed computing, spark.

# Introduction

Particle physics is a branch of physics that explores the fundamental particles and forces that make up our universe. It is a complex field that requires large-scale experiments, sophisticated data analysis techniques, and powerful computing systems. The discovery of new particles and their properties is essential for advancing our understanding of the universe, and this has been a major goal of particle physics for several decades. Whereas, Particle discovery is a complex and challenging task, and the datasets involved in this field are massive and complex. The data generated by experiments in particle physics are typically high-dimensional and require specialized techniques for data analysis. These datasets are difficult to store, process, and analyze using traditional computing systems, and this has led to the development of specialized hardware and software solutions for particle physics data analysis. In recent years, there has been a growing interest in using machine learning techniques to analyze particle physics datasets. Machine learning algorithms can be used to identify patterns in large and complex datasets, and they have been shown to be effective in solving a wide range of problems in particle physics.

In order to investigate the fundamental nature of matter and the structure of space and time, physicists focus on simple interactions, such as the collision of subatomic particles at high energies[1]. Particle accelerators provide a means to observe the subatomic particles produced by these collisions and study their properties. However, the experimental measurements obtained from these collisions often lack precision. As a result, machine learning (ML) techniques have become increasingly important in the analysis of data from particle physics experiments. Standardized ML software packages are commonly used by the research community to extract significant features from the raw data, thereby improving the statistical power of the analysis.

The Higgs boson particle is a fundamental particle and the final missing piece in the standard model of particle physics. As explained in [2] the standard model sets the rules for subatomic particles and forces, and postulates that elementary particles are massless at high energies but can acquire mass at low energies. The mechanism behind this mass acquisition remained a mystery in theoretical physics for a long time. In 1964, Peter Higgs and others proposed a mechanism that involved a field, now known as the Higgs field, which particles can interact with to acquire mass. The Higgs boson is the associated particle of the Higgs field, and its detection is of great importance in the field of particle physics.

The Higgs boson can decay in various ways, or channels, producing different particles. As described by [1] the decay of the Higgs boson into fermion pairs, such as tau leptons (𝜏) or b-quarks, is of particular interest as it would provide evidence that the Higgs field also gives mass to fundamental fermions. Among the available decay modes, the decay to a pair of tau leptons is the most promising, with a manageable background and a modest branching ratio. The detection of Higgs boson was a major achievement for experimental particle physics, and it was accomplished through the use of large particle accelerators, such as the Large Hadron Collider (LHC) at CERN. The LHC collides protons at very high energies, producing a large number of subatomic particles, including Higgs bosons[3].

The detection of the Higgs boson required the analysis of vast amounts of data, which was made possible through the use of machine learning techniques. ML algorithms were used to filter out the background noise and identify the signal corresponding to the Higgs boson decay into two pairs. This involved training ML models on simulated data, and then applying these models to the actual data collected from the LHC experiments. According to [4] The discovery of the Higgs boson has opened up new avenues for research in particle physics and has confirmed the importance of the Higgs field in the generation of mass for elementary particles. Ongoing studies of the Higgs boson and its properties continue to shed light on the nature of the universe at its most fundamental level.

However, the computational requirements of machine learning algorithms can be significant due to the amount of features required to be analysed and this has led to the development of specialized computing frameworks for accelerated machine learning. One such framework is the Python-based NVIDIA RAPIDS, which provides a suite of open-source libraries and APIs for accelerated machine learning. RAPIDS allows developers to leverage the power of NVIDIA GPUs to accelerate data preparation, model training, and inference, enabling faster and more efficient analysis of large and complex datasets[5].

This paper aims to explore the theoretical principles and objectives of accelerated machine learning using the NVIDIA RAPIDS framework. We will discuss the challenges associated with particle discovery and the role of accelerated machine learning . We will then present a detailed background on how particle discovery is undertaken currently, the challenges associated with data storage in this field and an overwiew of accelerated machine learning frameworks. We will then explore the potential of accelerated computing frameworks, such as RAPIDS, to provide a better solution for particle physics data analysis. We will discuss the benefits of parallel computing, distributed systems, Dask, and RAPIDS for accelerating machine learning algorithms. We will support these considerations with diagrams to aid in understanding.

Finally, we will present a detailed methodology for using the RAPIDS framework to analyze particle physics datasets. We will describe our data pre-processing techniques, chosen model architecture, hyperparameter selection, and evaluation methods. This research will evaluate the suitability of different processing architectures for specific computational tasks, such as CPU/GPU, and demonstrate a deep understanding of relevant RAPIDS ML concepts and techniques.

# Background

In recent times, machine learning algorithms have garnered significant attention as a promising approach for analyzing particle physics data. These algorithms offer the potential to identify patterns and extract valuable insights from large datasets that may not be easily discernible through traditional analysis methods. The application of machine learning algorithms in particle physics research is an evolving field that has demonstrated promising results[6]. Several techniques, including deep learning, decision trees, and clustering algorithms, have been extensively researched and applied to particle physics data. Of these techniques, deep learning algorithms have shown great potential in discovering new particles and identifying rare events in large datasets. By leveraging neural networks, deep learning algorithms can learn complex patterns and relationships within data, enabling them to make more accurate predictions and classifications. Decision trees are useful for identifying relevant features within data and making predictions based on those features[7]. Clustering algorithms can help identify groups of particles that share common characteristics and may have similar physical properties. Studies have demonstrated the potential of machine learning algorithms in particle physics research. For example, researchers at CERN have used deep learning techniques to identify rare Higgs boson events in large datasets, which has contributed to a better understanding of this fundamental particle[2]. Similarly, clustering algorithms have been used to classify different types of particles based on their properties. Machine learning algorithms offer a promising approach to analyzing particle physics data, enabling researchers to extract valuable insights from large datasets that may be difficult to discern through traditional analysis methods[8]. With the continued development of these algorithms, they have the potential to accelerate scientific discovery and advance our understanding of fundamental particles and the universe.

The Large Hadron Collider (LHC), situated at CERN, is a cutting-edge collider that is capable of producing high-energy particles and generating vast amounts of data[3]. In particle physics, an event refers to the outcomes that occur immediately after a fundamental interaction between subatomic particles, taking place over a very short period of time and in a well-defined region of space[9]. In the LHC, swarms of protons are accelerated in both directions along a circular path at incredibly high speeds. These swarms intersect in the ATLAS detector, resulting in hundreds of millions of proton-proton collisions per second[1]. The sensors in the detector detect these events, generating a sparse vector of approximately 100,000 dimensions, which is similar to an image or speech signal in classical machine learning applications. During the feature construction phase, information such as particle type, energy, and 3D direction is extracted from the raw data. Additionally, a variable-length list of four-tuples is transformed into a fixed-length feature vector containing up to tens of real-valued variables. Some of these variables are utilized in a real-time multi-stage cascade classifier, known as the trigger, to discard most of the irrelevant events classified as background events[6]. The remaining selected events, approximately 400 per second, are then recorded onto disks by a large CPU farm, resulting in petabytes of data per year[6]. The majority of the saved events still represent known processes, or background events, which are mainly produced by the decay of exotic particles that are already known from previous experiments. The aim of the offline analysis is to identify a region in the feature space, known as the selection region, that generates a significant excess of events, or signal events, compared to what is expected from known background processes. Once the selection region has been determined, a statistical test is applied to assess the significance of the excess. If the probability that the excess was caused by background processes falls below a certain threshold, it indicates the discovery of a new particle.

However, the storage and processing of this data can be challenging, with traditional methods often proving insufficient. The sheer volume of data generated can be overwhelming, and traditional storage solutions are often unable to handle the scale and complexity of the data. As a result, there has been a growing interest in using accelerated computing frameworks, such as NVIDIA RAPIDS and Apache Spark, to improve the storage and analysis of particle physics data. These frameworks leverage the power of modern GPUs and distributed computing to overcome the limitations of traditional computing architectures, making them invaluable tools for particle physics research.

Accelerated computing frameworks, such as NVIDIA RAPIDS and Apache Spark, have emerged as promising solutions to address the challenges of processing and analyzing large particle physics datasets[10]. With the exponential growth of data in particle physics and other scientific domains, traditional computing systems have struggled to keep pace with the demand for efficient data processing and analysis. The application of accelerated computing frameworks in scientific research is a significant advancement that leverages the power of modern GPUs and distributed computing to overcome the limitations of traditional computing architectures. NVIDIA RAPIDS is a suite of software libraries and APIs optimized to run on GPUs, offering data manipulation, machine learning, and visualization capabilities to data scientists and analysts[10]. Similarly, Apache Spark is a popular distributed computing framework that provides a unified engine for distributed data processing, with APIs for machine learning and graph processing. These frameworks are particularly suited for handling large-scale datasets, making them invaluable tools for particle physics research[11]. By leveraging GPUs and distributed computing, data processing and analysis can be accelerated to enable real-time analysis. This can lead to faster scientific discoveries, as researchers can rapidly analyze and explore large datasets. For instance, the use of Apache Spark by CERN researchers to analyze data from the Large Hadron Collider resulted in the discovery of new particles and other important findings[12]. Overall, the application of accelerated computing frameworks, such as NVIDIA RAPIDS and Apache Spark, presents a promising approach to accelerating scientific discovery in particle physics and other fields. With their ability to handle large-scale datasets and perform real-time analysis, these frameworks offer significant potential for scientific progress in the future.

In summary, the analysis of particle physics data presents significant challenges due to the scale and complexity of the data. However, the development of advanced hardware and software solutions, such as accelerated computing frameworks and machine learning algorithms, has the potential to unlock new insights into particle physics and drive scientific progress.

# Considerations:

To address the challenges of particle discovery, accelerated computing frameworks such as RAPIDS provide a better solution. In this section, we discuss why and how such frameworks can be used to optimize the data processing and analysis.

RAPIDS is an open-source data science framework that leverages GPU acceleration to enable faster data processing and analysis. It provides a suite of libraries for data processing, machine learning, and graph analytics, including cuDF (GPU-accelerated data frames), cuML (GPU-accelerated machine learning), and cuGraph (GPU-accelerated graph analytics)[13]. The RAPIDS framework, is a Python-based platform designed for data science and machine learning.whereas CuDF is a GPU-accelerated data frame library that provides a pandas-like API, while cuML is a machine learning library that provides GPU-accelerated versions of popular machine learning algorithms[10]. One of the most significant benefits of using accelerated computing frameworks is the ability to use GPUs for data processing. GPUs have been found to be highly effective for data processing, providing faster processing times and increased efficiency compared to CPUs. RAPIDS leverages GPUs to accelerate the processing of large datasets and to enable faster machine learning model training and evaluation[14].

Apache Spark is an another open-source big data processing framework that enables faster data processing and analysis. [15] explains that it provides a comprehensive set of libraries for data processing, machine learning, and graph analytics, including Spark SQL (SQL-like interface for data processing), MLlib (machine learning library), and GraphX (graph analytics library). The Spark framework is designed to handle large-scale data processing and analysis using distributed computing, making it an ideal choice for big data applications. One of the key advantages of using Apache Spark is its ability to leverage distributed computing to speed up data processing. By distributing computation across multiple nodes in a cluster, Spark can handle large-scale data processing tasks more efficiently and effectively than traditional single-node processing. Additionally, Spark provides a wide range of APIs and libraries for data processing, machine learning, and graph analytics, making it a versatile framework for a variety of use cases. Another significant benefit of Apache Spark is its support for GPU acceleration. GPUs have been found to be highly effective for data processing and machine learning, providing faster processing times and increased efficiency compared to traditional CPU-based processing. Spark leverages GPUs to accelerate the processing of large datasets and enable faster machine learning model training and evaluation. The integration of GPU acceleration with Spark's distributed computing capabilities makes it an even more powerful tool for big data processing and analysis.

One of the main advantages of accelerated computing is the ability to perform parallel processing[5]. This means that data can be processed simultaneously by multiple processors, reducing the time required for analysis. RAPIDS supports parallel processing through the use of dask-ml,CUDA and cuDF libraries, which enable GPU acceleration. On the contrary, spark has its own ML libraires known as MLlib which provides a variety of different ML Algorithms, Featurization tools, Pipelines, Persistence and Utilities.

Distributed computing is essential for processing large datasets in a reasonable amount of time. Dask is a Python library that provides parallel and distributed computing capabilities, and it can be used with RAPIDS to accelerate data processing and analysis[16]. Dask has two main libraries for distributed computing, dask dataframes and joblib or just use distributed, which use scikit-learn to implement distributed computing[17]. This allows for distributed data processing, and it can speed up data analysis significantly. Spark is another popular framework for distributed computing, and it also supports distributed file systems like Hadoop Distributed File System (HDFS). Spark has integrated HDFS and Resilient Distributed Dataset (RDD) in its functions to enable distributed processing[18]. It uses map reduce, HIVE, and pig to retrieve or store data in the distributed network. In contrast to Dask, Spark has its own machine learning library, known as MLlib, which provides a variety of different machine learning algorithms, featurization tools, pipelines, persistence, and utilities. However, using Spark's proprietary machine learning libraries means that one may not be able to use pandas or similar Python libraries used for data pre-processing. Also, the model selection would be limited because of the availability of models in the library.

The utilization of accelerated computing frameworks, such as RAPIDS, to address the challenges of particle discovery can be limited by factors such as the availability of multiple nodes for distributed computing or the availability of GPUs for parallel computing. Furthermore, the suitability of the framework may depend on the stage of the project, whether it is for experimentation or production implementation. In an enterprise environment, the use of Apache Spark may be more feasible due to the ability to handle large amounts of data and compatibility with other Apache products. Additionally, the use of containerization can allow for the integration of distributed computing to improve the performance of Spark and enable parallel computing to be implemented separately using Dask.

It should be noted that the use of Spark may have limitations in terms of model selection and data pre-processing as it primarily uses its own proprietary machine learning libraries. In contrast, Dask can be opted for in experimentation stages where the amount of data is less than 1TB. For local experimentation without the need for production implementation, RAPIDS can be a viable option. Overall, the use of distributed computing can greatly improve the scalability and processing speed of accelerated computing frameworks. As shown in Figure 1, the utilization of Dask in conjunction with RAPIDS or Spark can enable the workload to be distributed across multiple nodes and allow for parallel processing, leading to improved performance and efficiencyDiagram

Description automatically generated

Figure 1(local and distributed network)

To illustrate the benefits of using accelerated computing, Figure 2 shows a comparison of parallel and serial computing. In this example, parallel computing can process the data more quickly by dividing the workload across multiple processors, while serial computing processes the data one task at a time. The use of accelerated computing frameworks enables parallel computing and can significantly reduce the time required for data processing and analysis.

Diagram

Description automatically generated

Figure 2 (Comparison of parallel and serial computing)

Moreover, parallel computing can be implemented in two ways firstly, if we have a distributed network we can use the resources from every system on network for processing data or we can use a GPU as it has multiple cores unlike CPU and it can process the data. This is illustrated in figure 3[19].

Chart

Description automatically generated

Figure 3(CPU and GPU structure)

In summary, accelerated computing frameworks such as RAPIDS offer a better solution for the challenges of particle discovery. The ability to perform parallel processing, use distributed systems, and leverage GPUs enables faster data processing and analysis, making it possible to analyze large datasets more quickly and accurately.

# Data description

The presented dataset has been generated via Monte Carlo simulations using official ATLAS full-detector simulation. The simulation consists of two parts: the first part simulates random proton-proton collisions based on our current understanding of particle physics, which reproduces the microscopic explosions resulting from such collisions. In the second part, the resulting particles are tracked through a virtual detector model, resulting in simulated events with properties that replicate the statistical properties of real events, while providing additional information about the collision process prior to particle detection. Specifically, the dataset includes a signal sample containing events in which Higgs bosons with a fixed mass of 125 GeV were produced, as well as a background sample generated by other known processes capable of producing events with at least one electron or muon and a hadronic tau, mimicking the signal. The background sample was derived from only three processes, namely the decay of the Z boson into two taus, events with a pair of top quarks that can have a lepton and a hadronic tau among their decay, and the decay of the W boson where one electron or muon and a hadronic tau can appear due to imperfections in particle identification.

The Higgs dataset used in this study comprises 11 million instances and 28 features, where the first column represents the class label, with 1 indicating signal and 0 indicating background. The remaining 28 features are kinematic properties of particles measured by the detectors in the accelerator, with the first 21 features representing low-level features such as lepton pT, lepton eta, lepton phi, missing energy magnitude, missing energy phi, and jet pt, eta, phi, and b-tag. The remaining seven features are high-level features derived by physicists to assist in distinguishing between the two classes. The dataset is publicly available from the UCI Machine Learning Repository, with the last 500,000 instances reserved for testing in previous studies. The research objective of this study is to explore the applicability of accelerated machine learning techniques utilizing the NVIDIA RAPIDS framework for analyzing the Higgs dataset, specifically examining the use of GPU-based processing and deep learning methods to eliminate the need for manual feature engineering by physicists.

Prior to analysis, Exploratory Data Analysis (EDA) will be conducted to identify any missing values, data imbalances, correlation matrices, and outliers. Additional information regarding each feature can be found in the original paper. The dataset has previously been utilized in other studies, including benchmark results utilizing Bayesian Decision Trees from a standard physics package and 5-layer neural networks. The competition-provided dataset includes information on 250,000 events, with each event containing 29 features (one float-type label column and 28 float-type features) and object-type target variable labels, as well as float-type variable weights.

The dataset was missing the labels of the features and were provided by [7]. The labels provided are as follows: lepton pT, lepton eta, lepton phi, missing energy magnitude, missing energy phi, jet 1 pt, jet 1 eta, jet 1 phi, jet 1 b-tag, jet 2 pt, jet 2 eta, jet 2 phi, jet 2 b-tag, jet 3 pt, jet 3 eta, jet 3 phi, jet 3 b-tag, jet 4 pt, jet 4 eta, jet 4 phi, jet 4 b-tag, m\_jj, m\_jjj, m\_lv, m\_jlv, m\_bb, m\_wbb, m\_wwbb[7].

Let’s perform the Exploratory Data Analysis(EDA), the data provided was not labelled hence we referred [7] to find the labels of our dataset and replaced them in the dataset provided and we produced labelled data. For the next step we checked for the missing values in our dataset this was carried out by analysing each column due to the large size of dataset[20]. It is not feasible to use pandas isna command for this purpose hence the suspected columns were searched for missing values yet there were no missing values in the dataset.

Followed by the process of checking the class distribution. The class column consists of the information regarding background and signal information we will classify on the basis of this information. Hence it is necessary to check the number of values available for both classes to check class distribution. upon conducting analysis it was identified that values available for signal class are 5829122 and for background class are 5170877 this is also illustrated in figure 4.

Chart, bar chart

Description automatically generated

Figure 4(Class distribution)

Moving forward we need to check the correlation matrix. The correlation matrix will later on allow us to manipulate the dataset by allowing us to reduce the number of features. the correlation matrix of this dataset is illustrated in figure 5.

Graphical user interface, application

Description automatically generated

Figure 5(Correlation matrix)

# Methodology

This section outlines the methodology utilized in our study, encompassing data pre-processing, model selection, training, and evaluation. The initial step in the analysis involved conducting exploratory data analysis (EDA) to acquire valuable insights into the data. EDA involved identifying missing values, analyzing correlation matrices, and checking for class imbalance.

Prior to model training, the Higgs dataset underwent pre-processing to ensure it was ready for the training phase. The pre-processing involved grouping of features, removing low and very high correlation, and negatively related features from the dataset. Additionally, the data type for mean squared error was adjusted, and the features were scaled to ensure uniformity. Pre-processing is necessary to guarantee consistency and eliminate bias in the data. The grouping of features it was evident from the dataset the values are generated from jet and there are a total number of four jets described in the dataset hence we had to divide the data by grouping the values of the jets this generates four groups of data. Moving forward the analysis of features was performed in which we can remove all the features which have a negative because they are not related at all, the values below 0.3 because they are not playing any important role and the values above 0.89 because they are very highly related hence, we only need one of these features.

For the training phase, we split the dataset into 80% training and 20% validation sets. We used the training set to train the model and the validation set to tune the hyperparameters. During training, we optimized the model's hyperparameters using a gridsearch to identify the best hyperparamters may be the best way to ensure that the model achieved maximum performance. We used the mean squared error (MSE) as the loss function for training the model.

For clustering, multiple techniques are available to identify the data and link it to the respective class. The methods used in our study include k-means, DBSCAN, and agglomerative clustering. DBSCAN is a density-based approach to spatial clustering and has numerous advantages over K Means or Agglomerative Clustering. DBSCAN can identify clusters of any convex shape, while K Means focuses on finding centroids and associating data points with that centroid in a spherical manner. Moreover, DBSCAN is resilient to areas of low density.

We used NVIDIA RAPIDS framework, which includes libraries and APIs for accelerating machine learning computations on GPUs. The cuML library, which is part of the RAPIDS framework, was used to build the machine learning models. The cuDF library was used to manipulate and pre-process the data. And scikit-learn’s joblib to implement for distributed computing not require this at the moment because we are relying on the data from a single dataset instead of a continuous data-stream hence the main focus is to execute parallel computing in our system.

The current task at hand is to solve the regression problem, and we can use random forest, SVMs, GBDTs, etc. random forest is an ensemble of decision trees that can improve performance by reducing overfitting and increasing accuracy[21]. GBDTs are a general-purpose decision tree that can be used to improve performance[22]. Random forests are not just resilient to missing data but also noise[21]. SVMs, A model that locates the hyperplane that divides the data into the distinct classes in the most effective manner. SVMs are able to process data with a high dimension and perform admirably on linearly and nonlinearly separable sets of information[23]. While choosing a model, it is essential to take into account the dimensions of the dataset, the nuances of the connections between the variables, the degree to which the model can be interpreted, and the performance metrics that are prioritised. In order to figure out which strategy is the most effective for solving a particular regression issue, it might be essential to test out a few different models and evaluate how well they work. We looked at a number of different models to see which would be the best fit for this particular investigation. The Gradient Boosting Decision Tree (GBDT) method was chosen for this investigation because it has been demonstrated to provide state-of-the-art performance in a number of machine learning applications and is particularly well-suited for regression. Because of its higher performance on big datasets, we decided to utilise the XGBoost version of GBDT[24]. In addition, dask already has a library for XGBoost, which is convenient given that dask is typically used for processing large datasets. Initially, DASK will configure XGBoost's master process to run on the DASK scheduler and Xboost's worker processes to run on the DASK worker processes. Second, it transfers all of the Pandas dataframes that are contained within the DASK dataframes to XGBoost and then it allows XGBoost to train. All of this is able to take place in the same process without the need for any data to be sent since XGBoost provides a great Python interface. Both of the distributed services are able to collaborate on the same data simultaneously. After the training for XGBoost is complete, DASK will finish cleaning up the XGBoost infrastructure, and then it will proceed as usual.

To evaluate the performance of the model, we used several metrics, including mean squared error (MSE), accuracy, precision, recall, F1-score and Confusion matrix. MSE is the average of the squared differences between the predicted and actual values. Accuracy measures the percentage of correct predictions, while precision measures the percentage of true positives out of all positive predictions. Recall measures the percentage of true positives out of all actual positive cases. F1-score is a weighted average of precision and recall. From confusion matrix we can further calsulate the sensitivity and specificity to analyse the classification of the model.

The NVIDIA RAPIDS framework was chosen because it enables the acceleration of machine learning computations on GPUs, which leads to faster training times and higher accuracy in an experimentation environment as we don’t want to use spark because of its complexity of implementation on a small scale project. The XGBoost implementation of GBDT was selected because it has been shown to achieve state-of-the-art performance in accelerated machine learning applications. The use of a grid search method and ride regression for optimizing hyperparameters helps ensure that the model achieves maximum performance. Finally, the selection of multiple evaluation metrics provides a comprehensive evaluation of the model's performance.

# references

[1] ‘The Higgs boson: a landmark discovery’, *ATLAS Experiment at CERN*. https://atlas.cern/Discover/Physics/Higgs (accessed Mar. 22, 2023).

[2] P. W. Higgs, ‘Broken Symmetries and the Masses of Gauge Bosons’, *Phys. Rev. Lett.*, vol. 13, no. 16, pp. 508–509, Oct. 1964, doi: 10.1103/PhysRevLett.13.508.

[3] ‘Large Hadron Collider project discovers three new exotic particles’, *Large Hadron Collider project discovers three new exotic particles*. https://www.manchester.ac.uk/discover/news/large-hadron-collider-project-discovers-three-new-exotic-particles/ (accessed Mar. 22, 2023).

[4] P. Baldi, P. Sadowski, and D. Whiteson, ‘Searching for Exotic Particles in High-Energy Physics with Deep Learning’, *Nat Commun*, vol. 5, no. 1, p. 4308, Jul. 2014, doi: 10.1038/ncomms5308.

[5] R. Merritt, ‘What Is Accelerated Computing?’, *NVIDIA Blog*, Sep. 01, 2021. https://blogs.nvidia.com/blog/2021/09/01/what-is-accelerated-computing/ (accessed Mar. 22, 2023).

[6] ‘ATLAS’, *CERN*, Mar. 03, 2023. https://home.cern/science/experiments/atlas (accessed Mar. 22, 2023).

[7] ‘UCI Machine Learning Repository: HIGGS Data Set’. https://archive.ics.uci.edu/ml/datasets/HIGGS (accessed Mar. 22, 2023).

[8] S. R. Ahmad, ‘Technical Report of Participation in Higgs Boson Machine Learning Challenge’. arXiv, Oct. 09, 2015. Accessed: Mar. 22, 2023. [Online]. Available: http://arxiv.org/abs/1510.02674

[9] ‘Higgs Boson Event Detection - Part 1. EDA’. https://kaggle.com/code/sugataghosh/higgs-boson-event-detection-part-1-eda (accessed Mar. 20, 2023).

[10] ‘RAPIDS | GPU Accelerated Data Science’. https://rapids.ai/ (accessed Mar. 22, 2023).

[11] ‘Apache Spark Tutorial with Examples’, *Spark By {Examples}*. https://sparkbyexamples.com/ (accessed Mar. 22, 2023).

[12] ‘Welcome to CERN Sparks! | sparks.web.cern.ch’. https://sparks.cern/ (accessed Mar. 22, 2023).

[13] ‘RAPIDS AI – Medium’. https://medium.com/rapids-ai (accessed Mar. 22, 2023).

[14] ‘NVIDIA RAPIDS Tutorial: GPU Accelerated Data Processing’. https://www.exxactcorp.com/blog/Deep-Learning/nvidia-rapids-tutorial (accessed Mar. 22, 2023).

[15] ‘Apache SparkTM - Unified Engine for large-scale data analytics’. https://spark.apache.org/ (accessed Mar. 22, 2023).

[16] ‘What is a Distributed System? - An Introductory Guide | UK’, *Confluent*. https://www.confluent.io/en-gb/learn/distributed-systems/ (accessed Mar. 22, 2023).

[17] ‘Dask | Scale the Python tools you love’. https://www.dask.org/ (accessed Mar. 22, 2023).

[18] ‘RDD Programming Guide - Spark 3.3.2 Documentation’. https://spark.apache.org/docs/latest/rdd-programming-guide.html (accessed Mar. 22, 2023).

[19] ‘CPU vs GPU | Definition and FAQs | HEAVY.AI’. https://www.heavy.ai/technical-glossary/cpu-vs-gpu (accessed Mar. 22, 2023).

[20] ‘(1) New Messages!’ https://docs.rapidminer.com/latest/studio/operators/cleansing/missing/replace\_missing\_values.html (accessed Mar. 22, 2023).

[21] ‘What is Random Forest? | IBM’. https://www.ibm.com/uk-en/topics/random-forest (accessed Mar. 22, 2023).

[22] ‘GBDTs & Random Forests As Feature Transformers | by Abhay Shukla | Nerd For Tech | Medium’. https://medium.com/nerd-for-tech/gbdts-random-forests-as-feature-transformers-cd0fcf2be89a (accessed Mar. 22, 2023).

[23] R. Gandhi, ‘Support Vector Machine — Introduction to Machine Learning Algorithms’, *Medium*, Jul. 05, 2018. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (accessed Mar. 22, 2023).

[24] J. Brownlee, ‘A Gentle Introduction to XGBoost for Applied Machine Learning’, *MachineLearningMastery.com*, Aug. 16, 2016. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed Mar. 22, 2023).