Machine Learning Approaches for Predicting Dark Matter Halo Assembly

HoHim Lee[1, *]

[1] *School of Physics and Astronomy, University of Nottingham, Nottingham, NG7 2RD, UK*

(Dated: August 30, 2023)

N-body Simulations have been one of the most important ways to study the formation of galaxies and large-scale structure in the universe. Traditionally, we create merger trees to track the history of a dark matter halo using halo finders-tree builder algorithms like SUBFIND-D-TREES and ROCKSTAR-ConsistentTrees. In this research, we explore the possibilities of using machine learning algorithms to create merger trees for dark matter halos. The simulation used in this project is from the Three Hundred Project [1], which includes 324 large galaxy clusters, and we have selected clusters 1 and 8 to be used in this project. We create the training data by pairing up halos that have progenitor relationships, and then we add the same amount of negative examples to the training data. The machine learning models we focus on are Random forests and Neural networks. Each model has three variants: using all features with specific coordinates for location and velocity; using all features with relative location and velocity; and using the features selected by the Correlation-based Feature Selection (CFS) process. Both of our models successfully learned to generate merger trees that resemble the structure of the real merger tree for a specific halo.

I. INTRODUCTION

The curiosity of humans to discover the mysteries of the cosmos has never been satisfied. In ancient times, we observed the night sky with naked eyes, looking for the movement and position of stars. Building up stories and theories that were heavily influenced by cultures and religions. After the invention of the telescope in the early 17th century, we had our first glimpse into the scale and complexity of the universe, and most importantly, we discovered that Earth is not the centre of the universe. In modern days, we have incredibly advanced space telescopes like the newly launched James Webb [2], which allow us to observe much further into the early age of the universe than we have ever before. With the advancement of technology, we will continue to build on previous knowledge while pushing our understanding of the universe to new heights.

A. N-body Simulations

The large-scale structure of the Universe refers to the patterns of galaxies and matter on scales much larger than individual galaxies and groupings of galaxies. This includes Galaxy clusters, Superclusters, Dark matter, and Dark energy [3]. The formation of these large structures has been one of the most important research areas in astrophysics. One way to understand these structures is to directly observe them using advanced telescopes, but besides observation, it is impossible for astrophysicists to conduct experiments directly on these large structures, such as galaxies, as it requires a tremendous amount of particles and billions of years of evolution. However, with modern computing powers, it is possible to simulate the behaviour of a large number of particles by incorporating Newton's laws of motion and universal gravitation [3], this kind of simulation is called Astrophysical N-body Simulations.

The origins of N-body simulations in astrophysics can be traced back to 1941, when Erik Holmberg used a remarkably ingenious method to study the dynamics of galaxies. Using a system of 37 lightbulbs, each representing a galactic component, Holmberg was able to physically model the inverse-square law of gravitation by observing the amount of light received by each 'particle' [4]. The brightness of the lightbulb served as a proxy for gravitational force, allowing Holmberg to study the gravitational interactions within his simple galaxy model.

With the rapid advancement of computational power in recent decades, the scale and complexity of N-body simulations have increased dramatically. We have moved from lightbulbs and galvanometers to supercomputers, enabling us to simulate the interactions of millions to billions of particles. For example, the IllustrisTNG project, which includes three volumes and 18 simulations, these simulations not only include gravity but also a range of complex physical processes such as gas dynamics, star formation, and feedback from stars and black holes. The latest TNG project has simulated the evolution of a universe in a cube of about 1 billion light-years on a side, starting 12 million years after the Big Bang and running up to the present day [5]. These simulations offer astrophysicists a deeper understanding of cosmic evolution. They provide intricate details and dynamic representations, allowing scientists to investigate galaxy formation, dark matter halo structures, and cosmic gas distribution. As computational power continues its rapid growth, N-body simulations are set to expand in scale and complexity. This progress promises even deeper insights into the mysteries of our universe.

* pphxl1@nottingham.ac.uk

B. Halo Finders

A cosmological simulation generates a large amount of data, which mainly includes the properties of particles, such as position and velocity [6]. A simulation can have billions of particles representing dark matter, gas, and stars. When enough particles form together due to universal gravitation, there will be a higher-than-average density region, and this region is known as a halo [7]. These halos are primarily made up of dark matter and serve as the gravitational wells in which galaxies form and evolve. In order to identify and categorized these halos, an algorithm is used, and this special algorithm is a halo finder.

The first category is the Friends-of-Friends (FoF) halo finder. The FoF algorithm was introduced by [8], and it works by linking together dark matter particles that are closer than a certain distance to each other. If two particles are within this distance, they are considered "friends". This process continues iteratively, linking friends of friends, until the entire halo is identified. There are several halo finders that utilise the FoF algorithm, including GADGET [6] and AHF (Amiga's Halo Finder) [9].

The second category is the Spherical Overdensity (SO) halo finder. The spherical over-density algorithm [10] start from a density peak region and creates a sphere around it until the average density within the sphere drops to a specified threshold, usually a fixed multiple of the critical or mean density of the universe. The radius at which this condition is met is typically called the "virial radius," and the halo's mass is the total mass within this sphere. But by using this method, it is able to capture the full shape of some complex, non-spherical halos [11]. Therefore, Spherical Overdensity are often used with other halo-finding approaches like Friends-of-Friends.

The third category is the Phase-Space-based halo finder. "Phase space" refers to a space in which every possible state of a system is represented. In cosmological simulations, the Phase-Space-based halo finder uses a six-dimensional phase space for each particle, three dimensions correspond to its position in space (x, y, z), and the other three dimensions correspond to its velocity (vx, vy, vz). Similar to the Friends-of-Friends approach, which uses position space, a phase-space-based halo finder will link particles that are close in phase space. With the addition of velocity, it can identify halos and especially sub-halos better than other algorithms that only use the position space of a particle [7]. ROCKSTAR [12] is one of the most well-known phase-space halo finders, it identifies and constructs halos by connecting micro-halos in six-dimensional phase space and then hierarchically merging them.

C. Halo Merger Trees

After running a halo finder on a cosmological simulation, a catalogue of halos will be saved in snapshot format, and each snapshot represents a different time or redshift in the simulation. This catalogue contains information for each halo identified by the halo finder, for example, the catalogue produced by the AHF halo finder [9] includes 85 properties detailing the location, mass, velocity, progenitors, and other information of the halos.

During the simulation, halos will merge together and form a bigger halo. To understand and track the evolution of halos over time, astrophysicists use the merger tree [13]. A merger tree starts by comparing halo catalogues from consecutive snapshots. For each halo in a later snapshot, it tries to identify its progenitors (or parent halos) in the previous snapshot. The goal is to trace the lineage of each halo to its earliest. As a result, a merger tree provides a hierarchical structure that represents the merger history of a specific halo in the simulation. Each node in the tree is a progenitor, and the links between nodes represent their relationship.

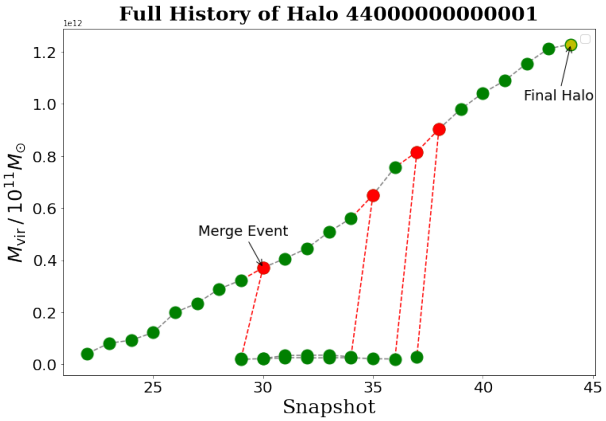

FIG. 1.

Fig. 1 shows a merger tree that is constructed from a catalogue of snapshots generated by the AHF halo finder [9]. This is from a simulation called The Three Hundred Project [1], which includes 324 large galaxy clusters that are re-simulated from the MultiDark-Planck Simulations [14]. From this merger tree, we can see that as the simulation time passes, the mass of the main halo keeps growing and has experienced four merger events, denoted by the red node and lines. This halo is taken from an earlier snapshot, and as the simulation goes on, there will be many more merger events for some big halos.

D. Machine Learning

Machine learning is a rapidly growing field within artificial intelligence that focuses on developing algorithms

and statistical models. These computational frameworks enable computers to perform tasks without explicit instructions. Tasks can span from image recognition and speech translation to complex predictive analytics. Machine-learning algorithms learn from data and make decisions or predictions based on it. By identifying patterns or structures within the data, machine learning enhances its knowledge [15]. There are three primary types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, models use labelled datasets to make predictions. Unsupervised learning uncovers patterns in unlabeled data sets, while reinforcement learning enables agents to learn through environmental interactions with reward-based feedback.

E. Project Aim and Report Structure

The aim of this project is to explore the feasibility of using machine learning techniques to predict dark matter halo mergers in cosmological simulations and generate a merger tree from the predictions. Halo mergers play an important role in the formation and evolution of galaxies and large-scale structures in the universe. Traditional methods for creating merger trees rely heavily on direct computations, which can be computationally expensive and time-consuming, especially when creating merger trees for halos in later snapshots. Machine learning, with its ability to learn complex patterns from large volumes of data, can be another way to effectively predict halo mergers and generate merger trees. We aim to predict halo progenitor relationships by training a model on a dataset of halo pairs with known progenitors. If the model indicates that a halo has multiple progenitors from the previous timestamp, it suggests a merger event has occurred. The predictions of this model could then be used to construct a merger tree, mapping out the complete merger history of halos. This approach has the potential to significantly improve the efficiency of merger tree construction, thus advancing our understanding of cosmic structure formation and evolution.

This study will proceed as follows: Section 2 will discuss some related research on halo merger tree construction and machine learning, which will help us understand how other researchers are currently creating merger trees. Section 3 will describe the methodology for how we preprocessed the data, analysed the data, and the architecture of the two models we created, which will be a random forest and a neural network. Section 4 will demonstrate the results, which will be separated into two parts, the first being the result of the model's performance in predicting Halo's progenitors. The second part will be the results of generating halo merger trees from the model's prediction. Finally, this study will conclude by discussing some future work that can be done in Section 5 and a conclusion in Section 6.

II. PREVIOUS WORK

The most significant research that is related to the goal of this project is titled "A deep learning approach to halo merger tree construction" [16] by Robles et al. (2022). This research also has the goal of constructing halo merger trees using machine learning approaches. In this research, the team primarily focused on the use of Generative Adversarial Networks (GANs) [17] to generate merger trees.

Generative Adversarial Networks (GANs) is a machine learning model introduced by Goodfellow et al. (2014) [18]. At it's core, GANs consist of two parts: the generator and the discriminator. The generator's main goal is to generate data that is similar to the real input data, whereas the discriminator's task is to distinguish between actual and generated data. During training, the generator will keep improving the accuracy of the generated data until the discriminator cannot distinguish the generated data from the real input data. This process will be repeated until the generator produces high-quality data that is indistinguishable from real data.

In the paper, the team trained their GAN model with real merger trees from the Evolution and Assembly of GaLaxies and their Environments (EAGLE) simulation suite [19]. SUBFIND–D-TREES and ROCKSTAR–ConsistentTrees are the two halo finder-tree builder algorithms that are used to construct the input merger trees. At the end of the paper, the team concludes that their GAN-based network has the ability to construct merger histories of low and intermediate-mass halos.

While Generative Adversarial Networks is proven to be effective at generating halo merger trees, but it also require constructing real merger trees from halo finder-tree builder algorithms, which can be computationally expensive, therefore, this project will take a simpler approach. Unlike using the whole merger tree as input to train a machine learning model, we use pairs of halos along with their properties like location, velocity, and mass. This pair of haloes will be labelled as a progenitor pair or not. We then use this training data to train a random forest and neural network model, which will be discussed in the following section.

III. METHODOLOGY

This section will focus on discussing the methods used while creating the machine learning models. This includes data exploration, data pre-processing, generating training data, feature selection, model selection, model architecture, merger trees, and testing.

A. Data Exploration

The Three Hundred Project is a cosmic simulation project that aims to provide a deeper understanding of the formation of large galaxy clusters through high-resolution simulations [1]. A total of 324 large galaxy clusters were produced in this project. These clusters were selected from the MultiDark-Planck Simulations [14] and re-simulated with full-physics hydrodynamical simulations. Each cluster simulation was run with Gadget-X and Gadget-MUSIC [20]. The evolution of the cluster is captured through 128 snapshots, covering a temporal span from redshift z = 17 to z = 0. The AHF halo finder [9] analysed all of the clusters and generated a set of halo catalogues for each cluster that are ordered by snapshots and include the progenitors information for each halos.

The outputs from the AHF halo finder are categorised into two primary file types: .AHF_halos and .AFH_mtree. The .AHF_halos files are the main data files, containing 85 distinct properties per halo. The .AFH_mtree files are dedicated to lineage data, offering detailed progenitor information, which is essential for understanding the formation of each halo.

Considering the extensive nature of each cluster, this research will only focus on two clusters: clusters 1 and 8. Appendix C shows the visualisation of both clusters simulated using Gadget-MUSIC [20]. The idea of using two clusters is that we can cross-validate our results, which can ensure our models not only work on one cluster but also have the ability to generalise to other clusters.

Here, we performed some basic data analysis on the 128 .AHF_halos files of cluster 1. We first converted all the files into CSV files for ease of data manipulation. We then used Python libraries such as Pandas and Matplotlib to visualise and explore the data. In total, there are 85 columns for each file, but according to the AHF halo finder user guide, the first 43 columns are integral properties, and the columns after that are radial profiles of selected properties. Considering we are only interested in the features that directly relate to the halo, we will only keep the first 43 features. Detailed description of each feature is shown in appendix A. Upon initial inspection of all the files, we found out that the virial mass of halos varies significantly, with a median value of 2.34×10^{10} and a maximum value of 3.85×10^{11}. This is expected because, at the early stage of the simulation, most of the halo will have a lower mass as they just start to form, but at the later stage, there will be some massive halos with much larger mass as they keep merging with other smaller halos. The kinetic and potential energies vary across halos, with potential energy typically being negative and having larger absolute values than kinetic energy. Most of the coordinates of the halos are distributed around a central point, and the velocities of the halos vary around zero, indicating random motion in different directions.

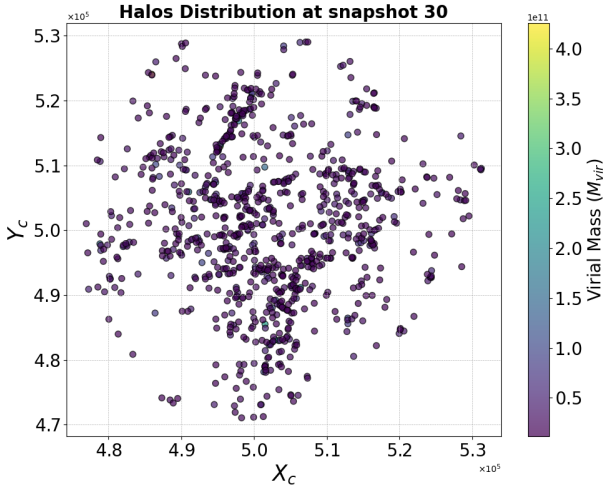

FIG. 2.

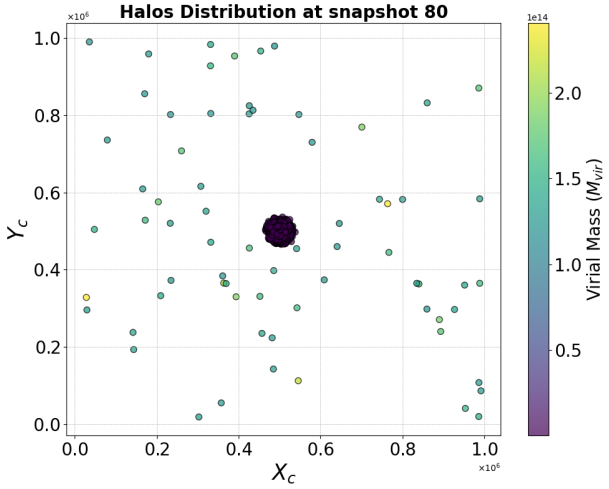

FIG. 3.

Fig 2 and fig. 3 show the spatial distribution of halos in the X and Y planes, with the colour gradient representing the virial mass. We can see that at snapshot 30, the majority of halos are distributed across the simulation space and have lower mass, but as the simulation progresses, the halos are clustered at a central point.

After visualising the distribution of the locations, we also want to understand the dynamical states of the halos. To do this, we calculate the magnitude of the velocities, which is done by:

$$\sqrt{VXc^2 + VYc^2 + VZc^2} \tag{1}$$

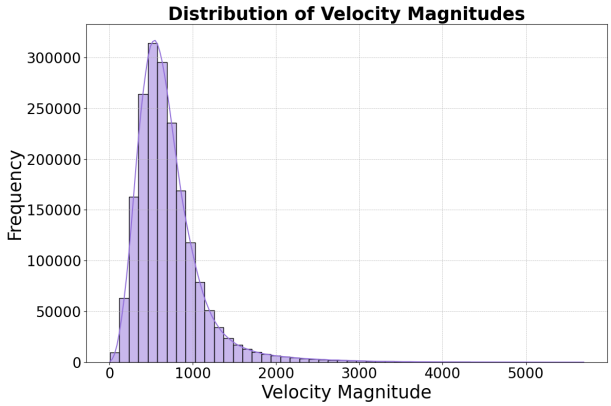

FIG. 4.

Fig 4 shows the distribution of velocity magnitudes of all the halos in cluster 1. We can see that most halos have velocity magnitudes clustered around a particular range, which can be viewed as the "typical" speeds of halos in this simulation. The distribution shows a long tail shape, indicating that there are some halos with exceptionally high velocities. These could be halos that recently underwent significant gravitational interactions or are in the process of merging.

B. Data Pre-Processing

As we now have a deeper understanding of the distribution of the halos, we continue to preprocess the data. We do this by first converting all the m_tree files into CSVs. The m_tree files contain the progenitors information of the halos, so it will be important that we extract this data from all the m_tree files and combine it with the corresponding halo in the halos snapshot files. We do that by processing all the snapshot files for both clusters 1 and 8, merging them with the progenitor information based on a common 'ID' column, and we also added two columns for each file: the redshift and snapshot number.

After we check for missing values in the files, the columns 'numProgenitors' and 'ProgenitorsID' include missing values. This is because not every halo will have progenitors, and this happens when a halo is newly identified by the halo finder in that snapshot and does not exist in the previous snapshots.

After we process all the data, we have a total of 128 combined snapshot files for each cluster, which include the progenitors information and the two added columns.

C. Training Data

The aim of our machine learning models is to predict whether or not a pair of haloes shares a progenitor relationship, and by predicting all potential progenitors for a given halo, we can construct halo merger trees. This will be a binary classification problem, as we only need the answer of is progenitor or is not progenitor. Before we can train a model capable of doing so, we need to generate a training file from which the models can learn. The structure of the training data will be as follows: Each row of the training data will represent a pair of halos, and that row will also contain all of the features for both halos. We will also be adding three additional features to each pair of halos, which will be the relative position, relative velocity, and target variable "is-progenitor".

In order to generate a training file like that, we break this down into three steps. First, an algorithm will process the combined snapshot files to generate positive cases. The algorithm will compare the relationships between the halos from two consecutive snapshots and filter out the pairs of halos that have progenitor relationships. Then it computes the relative location and velocity of these pairs in 3D space using the Euclidean distance formula 2. Due to limited computational resources, we limit the maximum number of halos to process per snapshot to 10,000 and randomly select 3500 pairs. This ensures we have examples across all the snapshots while our hardware can handle them.

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{2}$$

In the second step, we use another algorithm that is similar to the first one to generate negative cases. Instead of filtering out the pairs of halos that have progenitor relationships, it searches for pairs of halos that are not progenitors or descendants of each other. We also set a maximum distance for non-progenitor pairs. This is to encourage the models not to overrely on the location of the halos but rather to incorporate other features like the velocity to make predictions. The maximum distance is adaptive, which starts at 500 at the early snapshots and goes up to 5000 at the later snapshots. This is to accommodate the situation that smaller-mass halos can merge when closer together, while a more massive halo can attract a lower-mass halo from a greater distance.

The final step in generating a training file is to combine the positive cases with the negative cases to create a complete training dataset. After the data is combined, we shuffle and balance the data to have the same amount of positive and negative examples. This ensures data randomness, thereby preventing biases and improving the generalisation ability of the machine learning models. Finally, we have successfully produced a training dataset with 658908 pairs of halos, each with 97 columns, that is ready for use to train a machine learning model.

D. Features Selection

After we have generated the training dataset, we move on to take a deeper look at each of the features. Now that the training has a total of 97 features, we want to explore which features have the highest contribution when predicting the target variable "Is_Progenitor". Therefore,

we performed features selection with two different approaches: Feature Importance from Tree-based models and Correlation-based Feature Selection (CFS) [21].

Features selection, also known as variable selection, is the process of selecting a subset of relevant features for use in model construction, and it comes with some benefits when training a machine learning model. First, it can reduce the training time of the models. With fewer features, the amount of data the model's algorithms need to process is reduced, resulting in a faster training time. Second, it can avoid or reduce the effect of the Curse of Dimensionality [22], which is a problem in machine learning with high-dimensional data. The amount of data required to accurately generalise the machine learning model grows exponentially as the number of dimensions or features increases. By reducing the number of features, we can lower the risk of the Curse of Dimensionality. Features selection can also remove redundant features by looking at the correlation between features and removing the ones with a high correlation with each other as they provide redundant information.

The first approach we use to perform features selection is by using tree-based models like Random Forest and XGBoost. Tree-based models determine feature importance by evaluating how often a feature is used to split the data and the improvement it brings to the predictions. The importance of a feature is calculated as the cumulative reduction in a model's error or increase in purity in terms of classification. Features that frequently lead to higher purity or more accurate splits will have a higher importance score, which represents the relative contribution of each feature to the prediction. Here, we implemented this approach by using a Random Forest model. Table I shows the features with the top 10 importance scores. We can see that relative location and velocity made up around 60% of the prediction, which suggests they are the two most important features in the training data.

No.	Feature	Importance
1	rel_location	0.347837
2	rel_velocity	0.264440
3	snapshot	0.084687
4	progenitor_redshift	0.067329
5	progenitor_snapshot	0.055632
6	redshift	0.038609
7	numProgenitors	0.016833
8	progenitor_v_esc	0.006697
9	Epot	0.005841
10	Mvir	0.005779

TABLE I. Feature Importances from Tree-based Model

To compare our feature importance results, we use Correlation-based Feature Selection (CFS) [21]. This approach not only compares the correlation of the features to the target variable but also to each other. As a result, it can select feature subsets that are highly correlated with the target variable while being uncorrelated with each other. This is done by computing the correlation matrix for the features and the target. Then select features that have a high correlation with the target at a determined threshold. From those selected features, exclude any features that are highly correlated with another determined threshold.

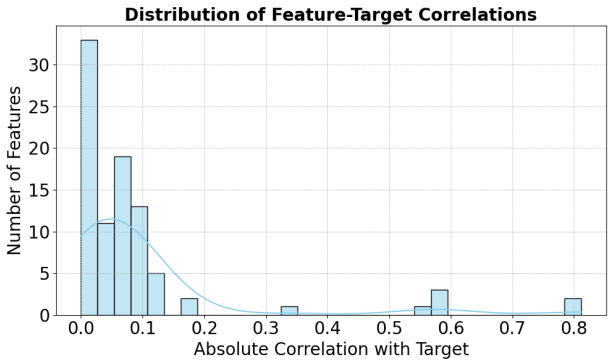

FIG. 5.

The distribution of the features and target variable correlations is shown in fig. 5. We can see from this histogram that a large number of features have correlations close to 0, indicating little to no linear relationship with the target, while a few features have strong correlations with the target variable. To ensure the threshold used to determine the features we select can include all the features that lead to the highest accuracy, we perform cross-validation with a range of pre-defined thresholds.

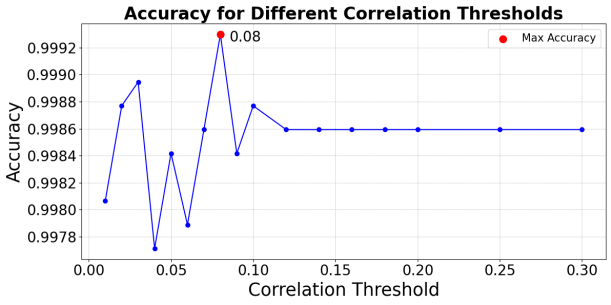

FIG. 6.

The result on fig. 6 shows that a threshold of 0.08 gives us the set of features with the highest accuracy. After filtering the features with 0.08 threshold, we create another correlation matrix that compares the features to each other, and then we remove redundant features that have a correlation score higher than 0.8. Table II shows the top 10 features selected by CFS, and from that, we can see it is similar to the top 10 features from table I, with relative location and velocity at the top, which suggests they are in fact the most important features in the training data. Appendix B has included all the features selected by CFS.

No.	Feature
1	rel_location
2	rel_velocity
3	progenitor_snapshot
4	progenitor_redshift
5	numProgenitors
6	cNFW
7	lambda
8	progenitor_lambda
9	Rmax
10	nbins

TABLE II. Top 10 Features from CFS

E. Model Selection

Machine learning models can be categorised into three types: supervised, unsupervised, and reinforcement learning. As we have the training data with correctly labelled targets, we will be using supervised learning. The objective of supervised learning models is to learn the mapping between input and output data and make predictions on new, unseen data based on the learned mapping. In order for the models to learn this mapping, we need to train them with labelled data, and during this training process, the model makes predictions and adjusts its weights based on the difference between its predictions and the actual outcomes, typically using a process like gradient descent. Supervised learning includes a wide range of machine learning models, from simple models like linear regressions and decision trees to complex architectures like deep neural networks. In this project, we will be using random forests and neural networks, which provide a good comparison between a simple and complex model architecture and allow cross-validation between two types of models.

1. *Decision Tree and Random Forest*

Algorithm 1 Binary Decision Tree for Classification

```
Initialize root node R with complete dataset.
Determine best feature for decision node.
Divide R using best feature and condition.
for each subset do
    Split data using using best feature.
end for
while stopping condition is not met do
    Repeat steps 2-4.
end while
```

A Decision Tree is a machine learning model that makes decisions based on asking a series of questions related to the features of the data, visually resembling a tree structure. In each decision tree, there will be two types of nodes: decision nodes and leave nodes. The decision node contains the condition to split the data, while the leave node contains the group of data and the associated label that share similar characteristics. Algorithm 1 shows the process of performing binary classification for a decision tree. There are two methods to determine the best feature and condition at each decision node: gini impurity and entropy. The formula for calculating entropy is:

$$H(S) = -\sum_{i=1}^{C} p_i \log_2(p_i) \tag{3}$$

Here, $H(S)$ is the entropy of data S, and p_i represents the probability of the i^{th} class in S, and the summation runs over all classes C. Then the Information Gain when split using feature A can be calculated by:

$$\text{IG}(S, A) = H(S) - \sum p(T) \times H(T) \tag{4}$$

where $\text{IG}(S, A)$ represents the information gain of data S using feature A. $H(S)$ is the entropy of the data S. $p(T)$ is the proportion of the subset T to S. And $H(T)$ is the entropy of the subset T. As for Gini Impurity, it is calculated by:

$$G(S) = 1 - \sum_{i=1}^{C} p_i^2 \tag{5}$$

and the Information Gain for Gini Impurity:

$$\text{IG}(S, A) = \sum_{T} p(T) \times G(T) \tag{6}$$

According to an experiment on Gini Impurity and Entropy with a decision tree, the performances of the two are very similar [23] with entropy slightly better but at the cost of a longer processing time. Due to limited competing resources, we will be using Gini Impurity to determine the data split.

Although Decision Tree have the advantage of high robustness on the scale of features and interpretability, they are also prone to overfitting as the tree gets deeper. To reduce this problem, we can use an ensemble of decision trees, which is a Random Forest. A Random Forest makes decisions based on the aggregation of different varieties of decision trees. This adds robustness to the model and therefore may leads to higher accuracy.

2. *Neural Network*

Also known as artificial neural networks (ANNs), Neural Network is a machine-learning algorithm that is inspired by the way neurons work in a human brain and has the ability to learn and generate predictions based on complex input data. A neural network consists of interconnected nodes, and each node will be a neuron. These neurons are categorised into three layers: an input layer, one or more hidden layers, and an output layer. The

way that a Neural Network learns from the input data is by adjusting the weight between each connected neuron during the training process using a weight adjustment algorithm like backpropagation.

Each neuron in the hidden layer and output layer contains an activation function. These functions introduce non-linearity to the model, which is essential for the model to learn complex data. The hidden layer and output layer can have different activation functions, but the output layer's activation function is task-dependent, for example, Sigmoid for binary classification and Softmax for multiclass classification or regression. In our case, we will be using Sigmoid for the output layer.

F. Models Architecture

After we have a deeper understanding of both Random forests and Neural networks, we move on to creating them. For both of the models, we first split the training data into training and test sets, with 20% allocated for testing and 80% for training. To have a consistent scale among features, we use the Standard Scaler to standardise them, ensuring they have a mean of zero and a variance of one.

To implement the Random forest model, we employ the Random Forest Classifier from the Scikit-Learn library. We have assigned a 20-time weight to class "1", which makes our target variable 20 times harder to be positive. This is to address some of the false-positive cases when using the model. We then train the random forest with the training set with the custom weight.

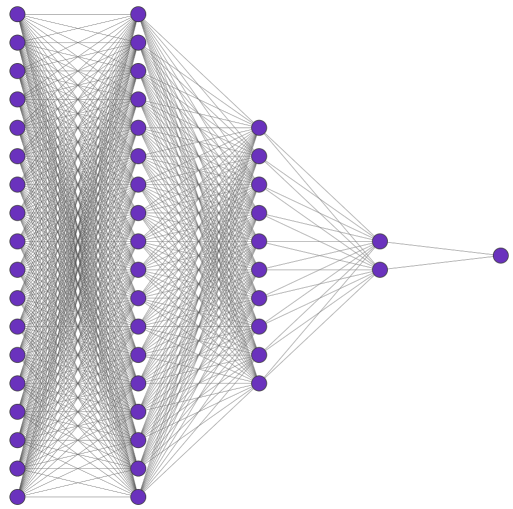

FIG. 7.

As for the Neural Network model, it is constructed using Keras' Sequential API. It begins with an input layer with the number of neurons matching the number of input features, followed by three hidden layers with 18, 10, and 2 neurons, respectively, all using ReLU activation. The output layer contains a single neuron with a sigmoid activation function for the binary classification task. The model is compiled with the binary cross-entropy loss function and the Adam optimizer, using accuracy as the evaluation metric. For training, the model uses a batch size of 32 and a total of 100 epochs, while setting aside 20% of the training data for validation. To enhance performance and reduce overfitting, the training process also employs callbacks for early stopping, which is based on validation loss with a patience of 5 epochs. Fig 7 shows a visual representation of the neural net model with 18 input features.

G. Merger Trees

The aim of the project is to generate merger trees for a halo, but now the output of our models is a binary output, which means each time the model can only predict one pair of halos. In order to utilise our models to predict the full merger history of a specific halo, we need a special algorithm that can incorporate the models to generate predictions repeatedly.

Algorithm 2 Create Merger Tree

```
function MergerTree(H, M, D)
    T ← [H]
    for each H′ in D[H['snapshot'] − 1] do
        if prediction from M on H and H′ is positive then
            Append H′ to T
        end if
    end for
    return T
end function
```

Algorithm 2 shows the high-level representation of the process of the algorithm. Where H is the current halo that we want to construct a merger tree for, M is the machine learning model used to predict relationships between halos, D is the dictionary containing all the snapshot files and T is the merger tree being constructed for the halo H. The algorithm starts by initialising the merger tree T with the current halo H, then it iterates over each halo H' from the snapshot prior to the current halo's snapshot. Using the machine learning model M, it predicts if the halo H' is a predecessor to the current halo. If the prediction is positive, H' is added to the merger tree T. By repeating the algorithm, we will be able to predict the complete history of a specific halo. The output of this algorithm will be a list that contains all the predicted progenitors of the target halo. This data can then be used to create merger tree plots.

H. Testing

In order to test how our models perform, we will be using the data from cluster 8 in the simulation, which was

No.	Model Type	Features
1	Neural network	All features with relative location and velocity
2	Neural network	All features with specific coordinates X, Y, Z for location and velocity
3	Neural network	With only selection features from CFS
4	Random forest	All features with relative location and velocity
5	Random forest	All features with specific coordinates X, Y, Z for location and velocity
6	Random forest	With only selection features from CFS

TABLE III. Comparison of Models and Their Features

not used during training and is from a different cluster. This ensures the models are being tested on unseen data to check their ability to generalise to new data. The testing data is generated in the same way as the training data described in section III C. In order to test how our feature selection performs, we have three variations for each model. Table III shows all the models with their details.

We will be testing our models using two approaches. The first is to test the model's performance on the binary classification task. The model's performance on the binary classification task does not completely reflect it's ability to generate merger trees because generating merger trees is a sequential process. If one prediction in the process is wrong, there is a great chance that the following prediction will be incorrect due to the model now using the wrong halo as a progenitor to make prediction. Therefore, for the second approach, we have selected 100 halos, which are from some early snapshots. We use our models to predict the history of these halos, which includes all the progenitors and merger events. By comparing the predicted and real histories of the halos, we will be able to determine the performance of the models at generating merger trees.

IV. RESULTS

A. Binary Classification

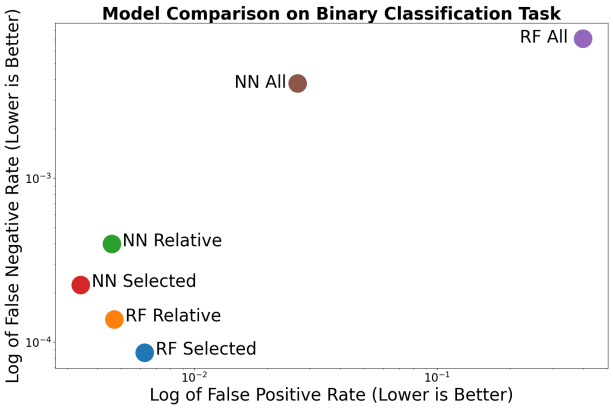

FIG. 8.

Fig 8 shows the testing results of all our models on the binary classification task. False Positive Rate (FPR) and False Negative Rate (FNR) are two metrics we used to compare the models. They provide in-depth insight into the model's performance, especially in binary classification tasks. False Positives in predicting halo progenitor relationships mean incorrectly identifying halos as having progenitor relationships when they don't, while False Negatives mean missing actual progenitor relationships between halos. By calculating the occurrence rate of these two metrics, we can compare the model's performance. A lower FPR in predicting halo progenitor relationships suggests fewer incorrect identifications of unrelated halos as having progenitor relationships. In contrast, a lower FNR indicates fewer missed actual progenitor relationships between halos.

1. *Neural Network with All Features (NN All)*

This model uses all 43 available features as input, with specific coordinates for location and velocity. This model has a higher false positive and negative rate than most of the others, only beating Random Forest with all features.

2. *Neural Network with Relative Features (NN Relative)*

Changing from specific coordinates to relative location and velocity for the pair of halos has led to a lower false positive and negative rate than the 'All Features' variant. This suggests that in the binary classification task, our neural network model favours relative features more than separate features for the coordinates of the halos.

3. *Neural Network with Selected Features (NN Selected)*

In this version of the neural network model, we only include the features selected by the Correlation-based Feature Selection (CFS) process. CFS has selected a total of 18 features. With only these features, the performance of the neural network is improved to a lower false positive and negative rate than the all-features and relative-features variants of the neural network model. This suggests that our feature selection process has been effective in retaining the most impactful features, allowing the model to achieve a well-rounded result.

4. *Random Forest with All Features (RF All)*

The first variant for the random forest model is the all features with specific coordinates version. Out of the six versions we have tested, this has the highest false-positive and false-negative rates. This indicates that Random Forest is not picking up the individual coordinates as well as the neural network does.

5. *Random Forest with Relative Features (RF Relative)*

With relative location and velocity, the performance of our random forest model has improved substantially. The false-negative rate was reduced compared to the NN Relative version, yet its false-positive rate remained similar to that of the NN Relative version. This suggests that the relative location and velocity of the pair of halos are the key features of the random forest model.

6. *Random Forest with Selected Features (RF Selected)*

While using only the 18 features selected by CFS in our random forest model, it achieved the lowest false-negative rate out of all the versions. As for the false-positive rate, it is slightly worse than the NN Relative and RF Relative versions. Like the NN Selected version, it shows feature selection can improve performance.

7. *Overall Comparison*

From all of the variations of the two models, our Random Forest with All Features version has the highest false-positive and false-negative rates. It is the worst-performing version in the binary classification task. Neural Network with Selected Features has the lowest false-positive rate, and Random Forest with Selected Features has the lowest false-negative rate. Random Forest with Relative Features provides a great balance between false-negative and false-positive rates while having good performance.

B. Merger Tree Construction

Based on the performance of the models on the binary classification task, we have selected the top four best-performing models to test with merger tree construction. These include the version with relative features and selected features for both the random forest and neural network. To test the model's performance on creating merger trees for halos, we compared the real history of a halo with the predicted history from our models. We primarily focus on the differences of progenitors and merge events between the two. Due to limited computing resources, we will only be able to test our models with the halos in early snapshots, as these halos have much fewer progenitors than the halos in later snapshots and require less time to process. The 100 target halos we selected for our models to generate merger trees are from snapshot 45 or earlier.

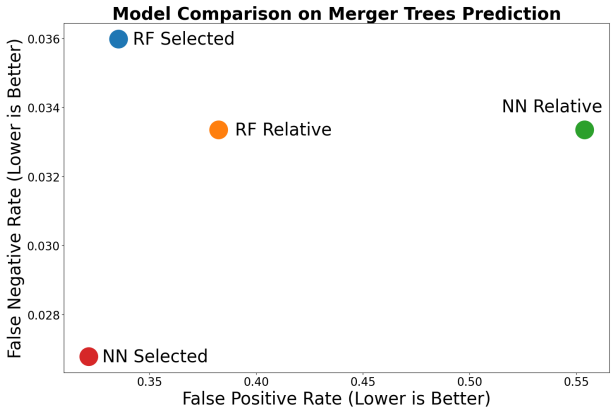

FIG. 9.

Fig 9 shows the performance of the four models on predicting halo progenitors histories. False positives indicate that the model wrongly predicts a halo as a progenitor of the target halo, and false negatives indicate that the model did not predict the actual progenitor of the target halo. Out of the four models, the neural network model with selected features has the lowest false positive and false negative rates.

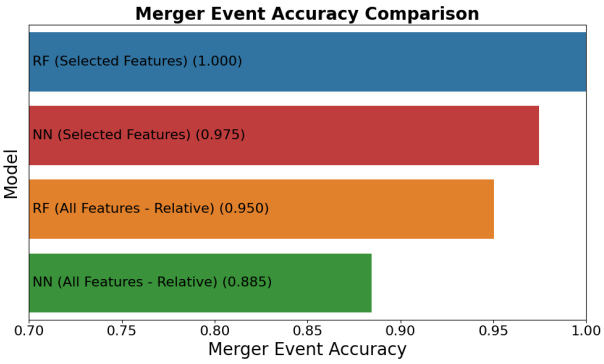

FIG. 10.

Fig 10 shows the model's accuracy in predicting merger events. Our Random Forest model with selected features has managed to predict all merger events for the 100 halos we selected. Other models also managed to predict more than 85% of the total merger events. Fig 18 and 19 shows the comparison between the merger tree generated by the random forest model using selected features to the merger tree that is constructed using the original data from the AHF halo finder.

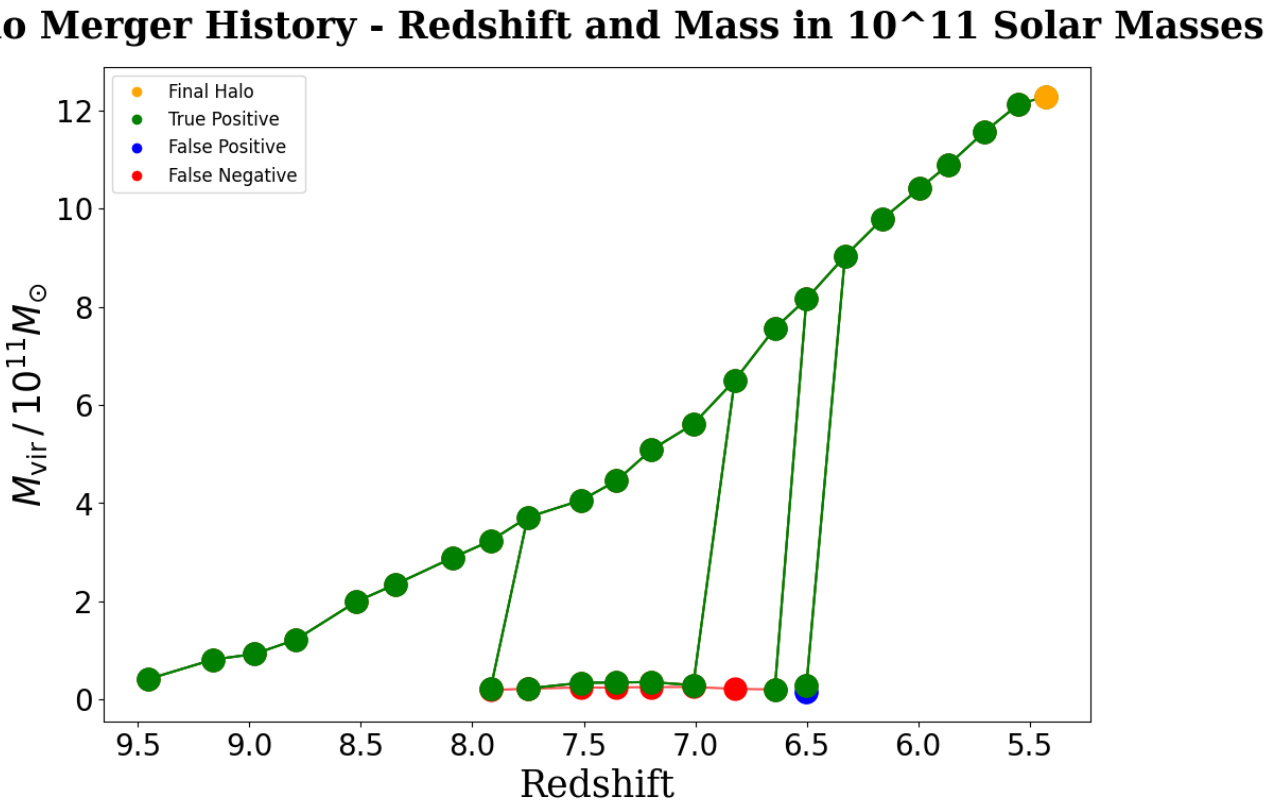

FIG. 11. Shows the comparison of the real and predicted merger tree of Halo 44000000000001 in Cluster 8. This is using the random forest model with selected features. It has successfully predicated the four merger events leading up to the target halo while having some false positives and negative progenitors. Refer to appendix D, E and F for more merger tree examples with different halos and models

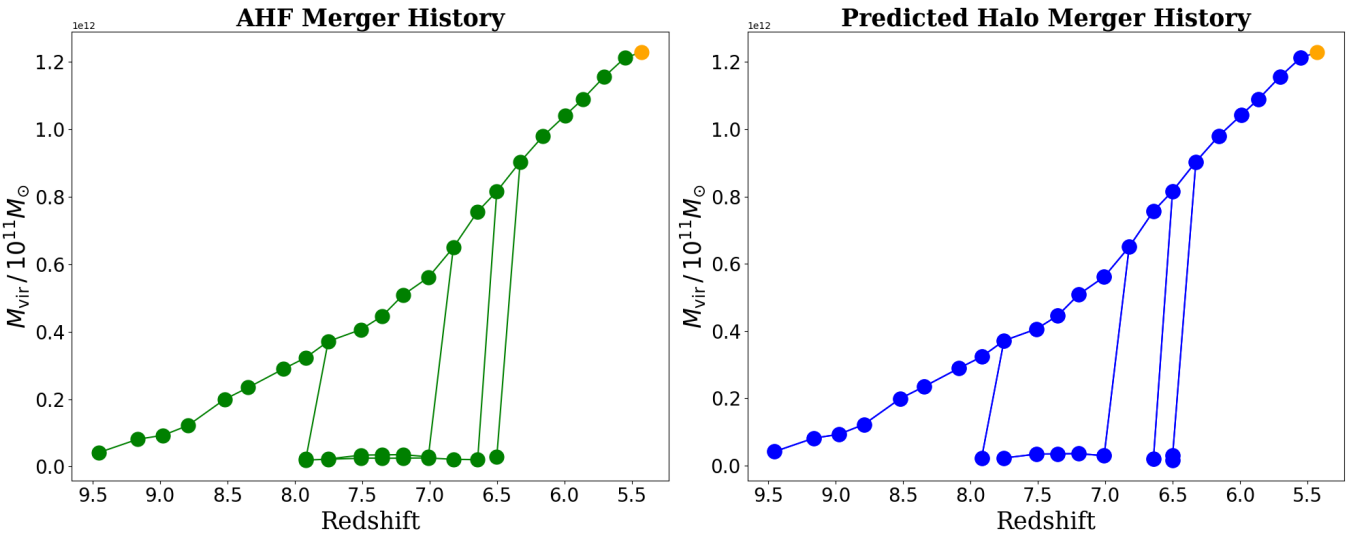

FIG. 12. Shows the real history of Halo 44000000000001 constructed from the AHF halo finder and the predicted history using random forest model with selected features.

V. RESULTS DISCUSSION

A. Uncertainties

Currently, the testing for merger tree construction is limited to 100 halos, which are randomly selected from snapshot 45 or earlier from cluster 8. This is due to the time required to generate merger trees for the halos in later snapshots is substantially longer than the halos in earlier snapshots, as the halos in later snapshots will have much more progenitors and merge events. Therefore, the results of generating merger trees from the models are only suitable to be used for halos in earlier snapshots and are relatively low-mass compared to halos in later snapshots. Also, all the testing, including binary classification and merger tree construction, is conducted using the data from cluster 8. As there are a total of 324 clusters in The Three Hundred Project [1], we are uncertain whether the results of this project will be suitable for other clusters or not.

B. Random Forest vs Neural Network

Our testing results show that both Random forests and Neural networks have the ability to trace back the history of a halo and create a merger tree from it. For the binary classification task, Random forests have a lower false negative rate and a similar false positive rate to the neural network. This can be due to a variety of factors. The first is the difference between the nature of the two machine learning algorithms. Random Forest is an ensemble learning method that makes use of multiple decision trees during training and can learn from a wide range of data. Also, it is not prone to overfitting, which potentially makes it less likely to miss positive cases, thus reducing false negatives. As for Neural networks, they have the ability to capture complex, non-linear relationships in the training data, but this can also lead to picking up noise in the training data. One possible explanation for the higher false positive rate than random forests is that our neural network models are picking up noise from some of the features, leading to potential false positives.

As for the merger tree's construction, our neural network model with selected features outperforms the random forest model in terms of both false positives and false negatives. Different from binary classification tasks, which only require a single prediction, creating a merger tree requires the model to recursively make predictions. Although random forests outperform neural networks in binary classification, in a more complex situation like generating merger trees, neural networks have the ability to capture the underlying pattern of the data better than random forests.

C. Selected features vs All features

From our model testing results, we can see that models using the selected features from the Correlation-based Feature Selection (CFS) have achieved better performance than models using all the features on both binary classification and merger tree construction. This shows that our feature selection process has the ability to increase models performance by focusing on the most relevant features and can retain the most significant features while discarding redundant or irrelevant ones. Here are some of the benefits we observed when using the selected features from CFS: The first benefit is improved accuracy when compared to using all of the features; this can be due to CFS successfully removing some irrelevant or redundant features that can introduce noise into the data. By removing these features, it decreased the likelihood that the models would become distracted by irrelevant data or features. Another benefit is that, with only the selected features, it significantly reduces the processing time required to train the model and construct merger trees.

D. Relative Metrics vs Absolute Metrics

In our binary classification testing, we have tested both random forests and neural network models using relative and absolute metrics. These metrics only affect the position and velocity of the pairs of halos, which are the primary features that our models used to determine the prediction. By converting the individual coordinates X, Y, Z to relative location and velocity for the pair of halos, we can see a substantial increase in performance with a lower false positive and negative rate on the binary classification task. We suggest that this result is due to relative metrics being better at describing the relationship between two halos than absolute values, which only describe the two halos in isolation. For machine learning modes, relative metrics like the relative velocity between two halos can provide insights into their interaction dynamics, which will be more relevant for predicting halo progenitor relationships than their absolute velocities in space. Another reason why the models favour relative values over absolute values is Scale Invariance. When determining the relationship between two halos, machine learning models do not necessarily need to know the exact location of the halos in the simulation box. The relative location captures what is more important, which is their distance from each other. This scale invariance can be beneficial for machine learning models, which can sometimes struggle with features that vary widely in scale. Relative metrics for location and velocity also served the purpose of dimension reduction. Without using individual coordinates, the features used to describe location and velocity have been reduced from six distinct features to only two. As discussed in the previous section, fewer features can also improve the model's accuracy.

E. False-Positives

Based on our testing results for both the random forest and neural network models, we found that the false positive rate and false negative rate are not balanced, especially when predicting halo histories to construct merger trees. Our models tend to overpredict, leading to a higher false-positive rate than the negative rate. This imbalance can be due to the fact that predicting progenitors on a large scale is an imbalanced task. Considering a snapshot that has thousands of halos, there will be only a few that will be the progenitor halo to the target halo in the next snapshot. But currently, our training data has the same number of positive and negative cases. The only technique we have implemented to combat this issue is to assign higher class weights to the positive class in the random forest models. One possible way to reduce this problem is to have a higher number of negative examples in the training data to create an imbalanced training data set. This will encourage the model to make fewer positive predictions, but this can also lead to a higher false-negative rate.

F. Binary Classification and Merger Tree

The process of generating a merger tree for a halo involves tracking the halo's history by repeatedly using the binary classification prediction results from the models. This is achieved by using our merger tree algorithm, which can pair every halo in a snapshot with the target halo and use the models to get the prediction. By repeating this process, we are able to track down all possible progenitors of a halo. Merge events are detected when the models predict that there is more than one progenitor for the specific halo in the previous snapshot. By using these two processes, we can create a complete merger tree for a halo while also identifying the merge events.

Looking at the results for the binary classification and merger tree construction tasks, we can see that the model's performance on binary classification does not reflect directly on merger tree construction. For example, in binary classification, our neural network model with relative features has a similar false positive rate to the one using only the selected features, but when tested on 100 merger trees, it has a much higher false positive rate than the selected features version, as well as being outperformed by the random forest model.

The first reason will be sequential dependencies. The construction of a merger tree is a sequential process. Even if the model achieved high accuracy in the binary classification task, a single error early in the sequence can alter and affect the entire tree. The cumulative error can grow as the sequence progresses, leading to a significantly flawed merger tree.

Second reason is that the merger tree accuracy is sensitive to false positives and negatives. In the process of creating merger tree, false positives (incorrectly predicting a progenitor) and false negatives (failing to detect a progenitor) can have significant implications for the tree's structure. For instance, a false positive could introduce a non-existent branch, while a false negative could miss an important merger event.

We agreed that while binary classification performance is important, the construction of a merger tree is influenced by a combination of the model's performance and other factors. Therefore, high performance in binary classification does not guarantee a perfect merger tree.

G. Effectiveness

The aim of this project is to investigate the effectiveness of using machine learning to predict halo mergers and create merger trees from the predictions. In the testing, we generated merger trees for 100 halos in early snapshots, and these halos typically have around 2 to 4 merge events. Both of our random forest and neural network models have managed to correctly predict more than 85% of the merger event. The random forest model with selected features has even successfully predicted all the merger events for 100 halos. Therefore, we agreed that we have partially achieved our aim, as we were not able to test the models for generating merger trees for the halos in later snapshots.

VI. FURTHER WORK

Due to the limited time frame and computing resources of this project, there are several experiments that we lack the time or resources to conduct. Currently, we have limited the number of positive halo pairs to 3500 pairs per snapshot. With more computing power, we can increase the number of examples in the training data. A complex machine learning model like a neural network requires a large amount of data to generalise well and avoid overfitting. By increasing the number of halo pairs in our dataset, we can potentially improve the accuracy and robustness of the model.

Also, with more computing resources, we could explore more complex and computationally intensive architectures for our neural network. Current constraints have led us to opt for simpler models, which, while efficient, might not capture all the intricacies of the dark matter halo assembly process.

Additionally, hyperparameter tuning, which is important for optimising the performance of machine learning models, especially neural networks, could be conducted on a more comprehensive scale. Presently, we've had to make do with limited grid searches or random searches for hyperparameters, but with more resources, we could employ techniques like Bayesian optimisation to find the optimal model parameters.

Another area of exploration could be the incorporation of temporal information into the model. Neural architec-

tures like Recurrent Neural Networks (RNNs) or Long Short-Term Memory networks (LSTMs) are designed to handle sequential data, which could be particularly relevant for tracking halo assembly over time. However, these models are also computationally more demanding, which makes them currently unfeasible for this project.

With more time and resources, we could also conduct thorough ablation studies to understand the importance of different features and components in our model. This would not only enhance the model's performance but also provide deeper insights into the underlying physical phenomena of dark matter halo assembly.

Testing is another area that requires more time and resources. The testing for merger tree construction is limited to only 100 halos, which are from snapshot 45 or earlier. We were unable to determine the model's effectiveness in generating merger trees for the halos in later snapshots, which have substantially more progenitors and merger events. Therefore, testing on more halos, ranging from all snapshots, is required to fully understand the effectiveness of the models. On a larger scale, we can expand the testing to other clusters in The Three Hundred project and possibly, adapt our model to test with other N-body simulation projects.

VII. CONCLUSIONS

In this project, we have explored the possibilities of using machine learning to predict the assembly of dark matter halos and to create halo merger trees. We start by exploring the data from a cosmological N-body simulation called The Three Hundred Project [1]. This simulation contains 324 large galaxy clusters. In this project, we used cluster 1 to generate training data and cluster 8 for testing. To train the machine learning model, we opt for a binary classification approach. In order to generate the training data for a binary classification task, we paired up halos that have progenitor relationships from two consecutive simulation snapshots, and each row in the training file represents a pair of halos with all their features. We also added two features, relative location and relative velocity, to each pair of halos.

After we created the training data from the raw data, we performed feature selection using two approaches: the first is to use feature importance from Tree-based models, and the second is Correlation-based Feature Selection (CFS). Both approaches suggest that relative location and velocity are the two most important features when predicting the progenitor relationship between two halos.

The machine learning models we focus on in this project are Random forest and Neural network. We have created three variants for each type of model, which include using all the features provided by the AFH halo finder, using relative values instead of absolute values for features, and using only the features that were selected by the CFS feature selection process. To compare the model's performance, we tested them on both the binary classification task and merger tree construction. A merger tree represents the full history of a halo, including all the merger events and all it's progenitors. We create merger trees by using an algorithm that uses the models to repeatedly predict the relationships between other halos and the target halo.

The merger tree testing we conducted only included 100 halos, which are from snapshot 45 or earlier in cluster 8. Therefore, the model's ability to create a merger tree for a larger mass halo in later snapshots is still unknown.

Our testing results show that both types of models have the ability to generate merger trees that resemble the actual evolution of a dark matter halo. Out of all model variants, the neural network with selected features performs the best in generating merger trees. Both types of models have successfully captured more than 85% of merger events. In conclusion, the models we created and trained in this project have the ability to predict the assembly history of low-mass dark matter halo using the snapshot data from the simulation, but further testing is required to investigate the performance of the models on some larger-mass halos.

[1] W. Cui, A. Knebe, G. Yepes, F. Pearce, C. Power, R. Dave, A. Arth, S. Borgani, K. Dolag, P. Elahi, R. Mostoghiu, G. Murante, E. Rasia, D. Stoppacher, J. Vega-Ferrero, Y. Wang, X. Yang, A. Benson, S. A. Cora, D. J. Croton, M. Sinha, A. R. H. Stevens, C. A. Vega-Martínez, J. Arthur, A. S. Baldi, R. Cañas, G. Cialone, D. Cunnama, M. De Petris, G. Durando, S. Ettori, S. Gottlöber, S. E. Nuza, L. J. Old, S. Pilipenko, J. G. Sorce, and C. Welker, The Three Hundred project: a large catalogue of theoretically modelled galaxy clusters for cosmological and astrophysical applications, **480**, 2898 (2018), arXiv:1809.04622 [astro-ph.GA].

[2] J. P. Gardner, J. C. Mather, M. Clampin, R. Doyon, K. A. Flanagan, M. Franx, M. A. Greenhouse, H. B. Hammel, J. B. Hutchings, P. Jakobsen, S. J. Lilly, J. I. Lunine, M. J. McCaughrean, M. Mountain, G. H. Rieke, M. J. Rieke, G. Sonneborn, M. Stiavelli, R. Windhorst, and G. S. Wright., The james webb space telescope, in *Astrophysics in the Next Decade*, edited by H. A. Thronson, M. Stiavelli, and A. Tielens (Springer Netherlands, Dordrecht, 2009) pp. 1–29.

[3] V. Springel, C. S. Frenk, and S. D. M. White, The large-scale structure of the universe, Nature **440**, 1137 (2006).

[4] E. Holmberg, On the Clustering Tendencies among the Nebulae. II. a Study of Encounters Between Laboratory Models of Stellar Systems by a New Integration Procedure., Astrophys. J. **94**, 385 (1941).

[5] D. Nelson, V. Springel, A. Pillepich, V. Rodriguez-Gomez, P. Torrey, S. Genel, M. Vogelsberger, R. Pak-

mor, F. Marinacci, R. Weinberger, L. Kelley, M. Lovell, B. Diemer, and L. Hernquist, The illustristng simulations: Public data release (2021), arXiv:1812.05609 [astro-ph.GA].
[6] V. Springel, The cosmological simulation code GADGET-2, **364**, 1105 (2005), arXiv:astro-ph/0505010 [astro-ph].
[7] A. Knebe, S. R. Knollmann, S. I. Muldrew, F. R. Pearce, M. A. Aragon-Calvo, Y. Ascasibar, P. S. Behroozi, D. Ceverino, S. Colombi, J. Diemand, K. Dolag, B. L. Falck, P. Fasel, J. Gardner, S. Gottlöber, C.-H. Hsu, F. Iannuzzi, A. Klypin, Z. Lukić , M. Maciejewski, C. McBride, M. C. Neyrinck, S. Planelles, D. Potter, V. Quilis, Y. Rasera, J. I. Read, P. M. Ricker, F. Roy, V. Springel, J. Stadel, G. Stinson, P. M. Sutter, V. Turchaninov, D. Tweed, G. Yepes, and M. Zemp, Haloes gone MAD72: The halo-finder comparison project, Monthly Notices of the Royal Astronomical Society **415**, 2293 (2011).
[8] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. M. White, The evolution of large-scale structure in a universe dominated by cold dark matter, Astrophys. J. **292**, 371 (1985).
[9] S. R. Knollmann and A. Knebe, AHF: Amiga's Halo Finder, **182**, 608 (2009), arXiv:0904.3662 [astro-ph.CO].
[10] W. H. Press and P. Schechter, Formation of Galaxies and Clusters of Galaxies by Self-Similar Gravitational Condensation, Astrophys. J. **187**, 425 (1974).
[11] R. Cañ as, P. J. Elahi, C. Welker, C. del P Lagos, C. Power, Y. Dubois, and C. Pichon, Introducing a new, robust galaxy-finder algorithm for simulations, Monthly Notices of the Royal Astronomical Society **482**, 2039 (2018).
[12] P. S. Behroozi, R. H. Wechsler, and H.-Y. Wu, The rockstar phase-space temporal halo finder and the velocity offsets of cluster cores, The Astrophysical Journal **762**, 109 (2012).
[13] C. Srisawat, A. Knebe, F. R. Pearce, A. Schneider, P. A. Thomas, P. Behroozi, K. Dolag, P. J. Elahi, J. Han, J. Helly, Y. Jing, I. Jung, J. Lee, Y.-Y. Mao, J. Onions, V. Rodriguez-Gomez, D. Tweed, and S. K. Yi, Sussing merger trees: The merger trees comparison project, Monthly Notices of the Royal Astronomical Society **436**, 150 (2013).
[14] A. Rodrí guez-Puebla, P. Behroozi, J. Primack, A. Klypin, C. Lee, and D. Hellinger, Halo and subhalo demographics with planck cosmological parameters: Bolshoi–planck and MultiDark–planck simulations, Monthly Notices of the Royal Astronomical Society **462**, 893 (2016).
[15] A. Jung, Machine learning: The basics (2022), arXiv:1805.05052 [cs.LG].
[16] S. Robles, J. S. Gó mez, A. R. Rivera, N. D. Padilla, and D. Dujovne, A deep learning approach to halo merger tree construction, Monthly Notices of the Royal Astronomical Society **514**, 3692 (2022).
[17] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, Generative adversarial networks: An overview, IEEE Signal Processing Magazine **35**, 53 (2018).
[18] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks (2014), arXiv:1406.2661 [stat.ML].
[19] J. Schaye, R. A. Crain, R. G. Bower, M. Furlong, M. Schaller, T. Theuns, C. D. Vecchia, C. S. Frenk, I. G. McCarthy, J. C. Helly, A. Jenkins, Y. M. Rosas-Guevara, S. D. M. White, M. Baes, C. M. Booth, P. Camps, J. F. Navarro, Y. Qu, A. Rahmati, T. Sawala, P. A. Thomas, and J. Trayford, The EAGLE project: simulating the evolution and assembly of galaxies and their environments, Monthly Notices of the Royal Astronomical Society **446**, 521 (2014).
[20] F. Sembolini, G. Yepes, M. De Petris, S. Gottlöber, L. Lamagna, and B. Comis, The MUSIC of galaxy clusters – I. Baryon properties and scaling relations of the thermal Sunyaev–Zel'dovich effect, Monthly Notices of the Royal Astronomical Society **429**, 323 (2012), https://academic.oup.com/mnras/article-pdf/429/1/323/3345321/sts339.pdf.
[21] N. Gopika and A. M. Kowshalaya M.E., Correlation based feature selection algorithm for machine learning, in *2018 3rd International Conference on Communication and Electronics Systems (ICCES)* (2018) pp. 692–695.
[22] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review, International Journal of Automation and Computing **14**, 503 (2017).
[23] P. Aznar, Decision trees: Gini vs entropy (2020).

Appendix A

No.	Feature/Properties	Details
1	ID	halo ID
2	hostHalo	ID of host halo, 0 (or -1) if halo itself is not a subhalo
3	numSubStruct	number subhalos inside halo
4	Mvir	mass of halo
5	npart	number of particles in halo
6	Xc	position of halo [kpc/h]
7	Yc	position of halo [kpc/h]
8	Zc	position of halo [kpc/h]
9	VXc	peculiar velocity of halo [km/sec]
10	Vyc	peculiar velocity of halo [km/sec]
11	VZc	peculiar velocity of halo [km/sec]
12	Rvir	virial radius [kpc/h]
13	Rmax	position of rotation curve maximum [kpc/h]
14	r2	position where r2 peaks [kpc/h]
15	mbp_offset	offset between most bound particle and halo centre [kpc/h]
16	com_offset	offset between centre-of-mass and halo centre [kpc/h]
17	Vmax	maximum of rotation curve [km/sec]
18	v_esc	escape velocity at Rvir [km/sec]
19	sigV	3D velocity dispersion [km/sec]
20	lambda	spin parameter (Bullock et al. 2001 definition)
21	lambdaE	classical spin parameter (Peebles' definition)
22	Lx	(orientation of) angular momentum vector
23	Ly	(orientation of) angular momentum vector
24	Lz	(orientation of) angular momentum vector
25	b	second largest axis of moment of inertia tensor [b/a]
26	c	third largest axis of moment of inertia tensor [c/a]
27	Eax	largest axis of moment of inertia tensor
28	Eay	largest axis of moment of inertia tensor
29	Eaz	largest axis of moment of inertia tensor
30	Ebx	second largest axis of moment of inertia tensor
31	Eby	second largest axis of moment of inertia tensor
32	Ebz	second largest axis of moment of inertia tensor
33	Ecx	third largest axis of moment of inertia tensor
34	Ecy	third largest axis of moment of inertia tensor
35	Ecz	third largest axis of moment of inertia tensor
36	ovdens	overdensity at virial radius
37	nbins	number of bins used for the *.AHF_profiles file
38	fMhires	mass fraction in high resolution particles for zoom simus
39	Ekin	kinetic energy [M€/h (km/sec)^2]
40	Epot	potential energy [M€/h (km/sec)^2]
41	SurfP	surface pressure (Shaw et al. 2006 definition) [M€/h (km/sec)^2]
42	Phi0	j0 (cf. unbinding procedure) [(km/sec)^2]
43	cNFW	NFW concentration (Prada et al. 2012 definition)

TABLE IV. All 43 features from the AHF halo finder

Appendix B

No.	Feature
1	rel_location
2	rel_velocity
3	progenitor_snapshot
4	progenitor_redshift
5	numProgenitors
6	cNFW
7	lambda
8	progenitor_lambda
9	Rmax
10	nbins
11	progenitor_nbins
12	progenitor_com_offset
13	r2
14	progenitor_SurfP
15	ovdens
16	SurfP
17	progenitor_cNFW
18	com_offset

TABLE V. The 18 selected features from correlation-based feature selection

Appendix C

(a) Cluster 1 at redshift z = 0.275

(b) Cluster 8 at redshift z = 0.220

FIG. 13.

Appendix D

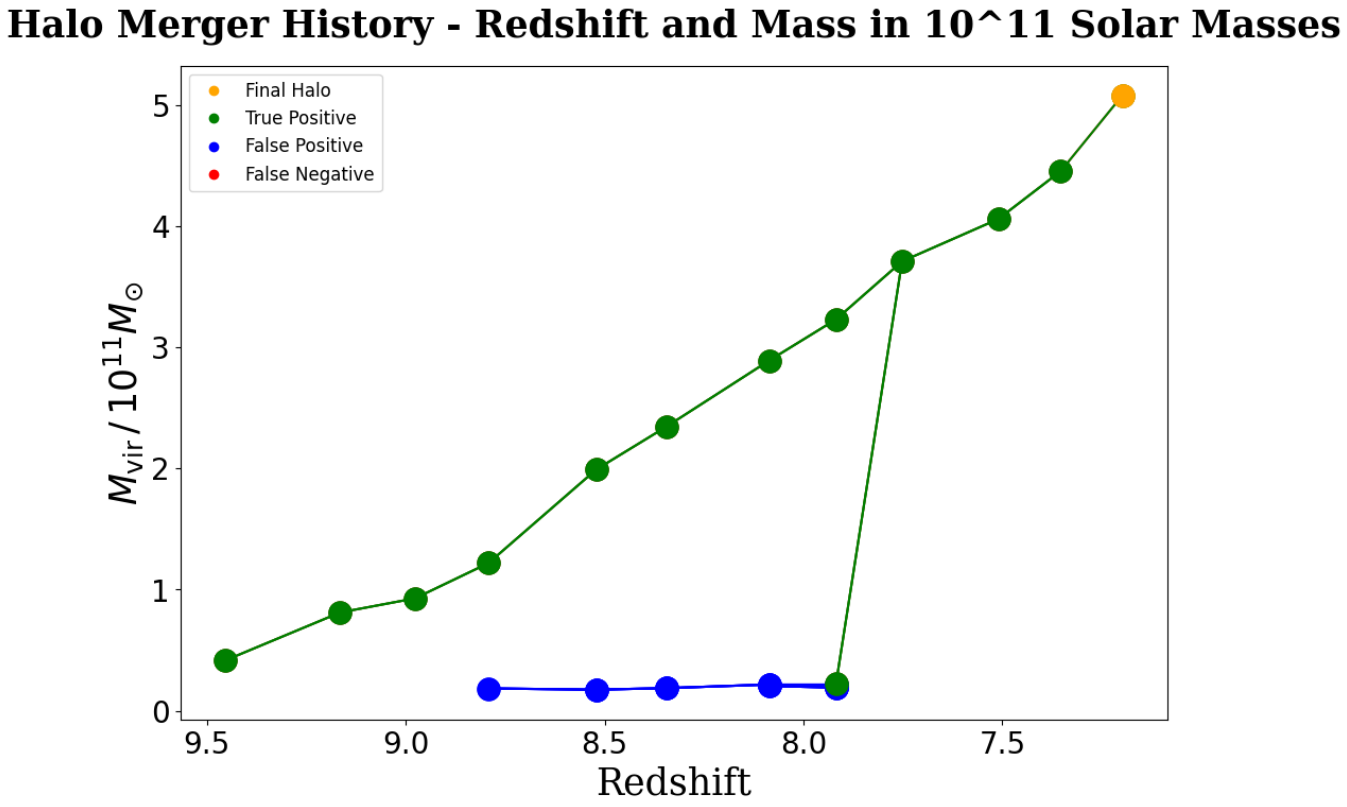

FIG. 14. Shows the comparison of the real and predicted merger tree of Halo 33000000000001 in Cluster 8. This is using the neural network model with selected features. It has successfully predicated the one merger events leading up to the target halo while having five false positives.

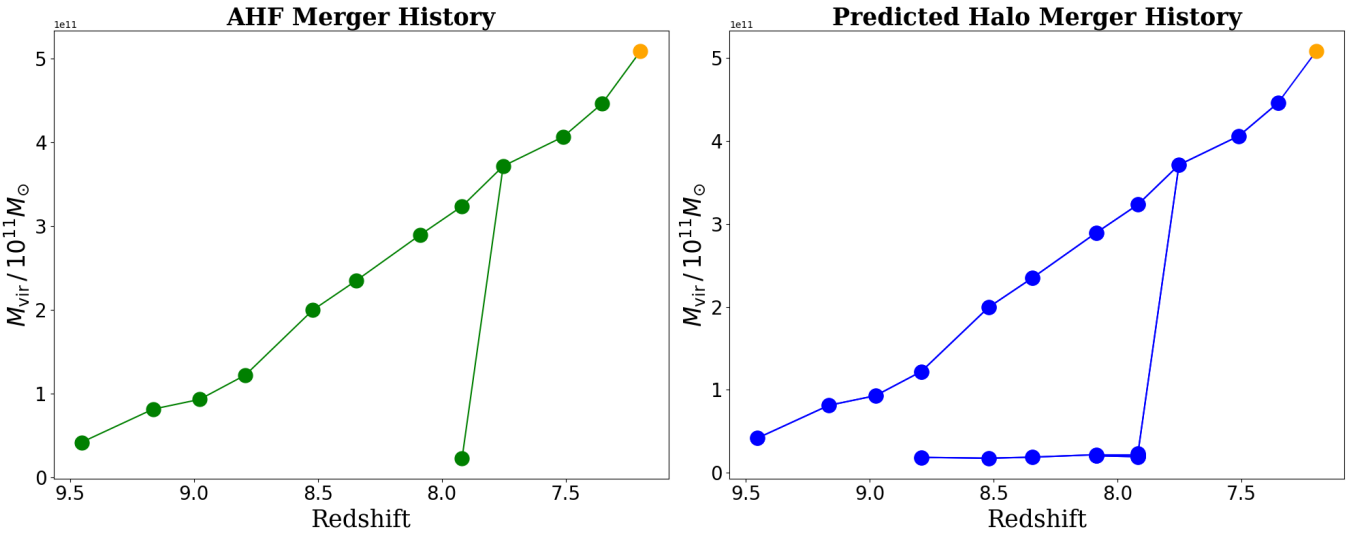

FIG. 15. Shows the real history of Halo 33000000000001 constructed from the AHF halo finder and the predicted history using neural network model with selected features.

Appendix E

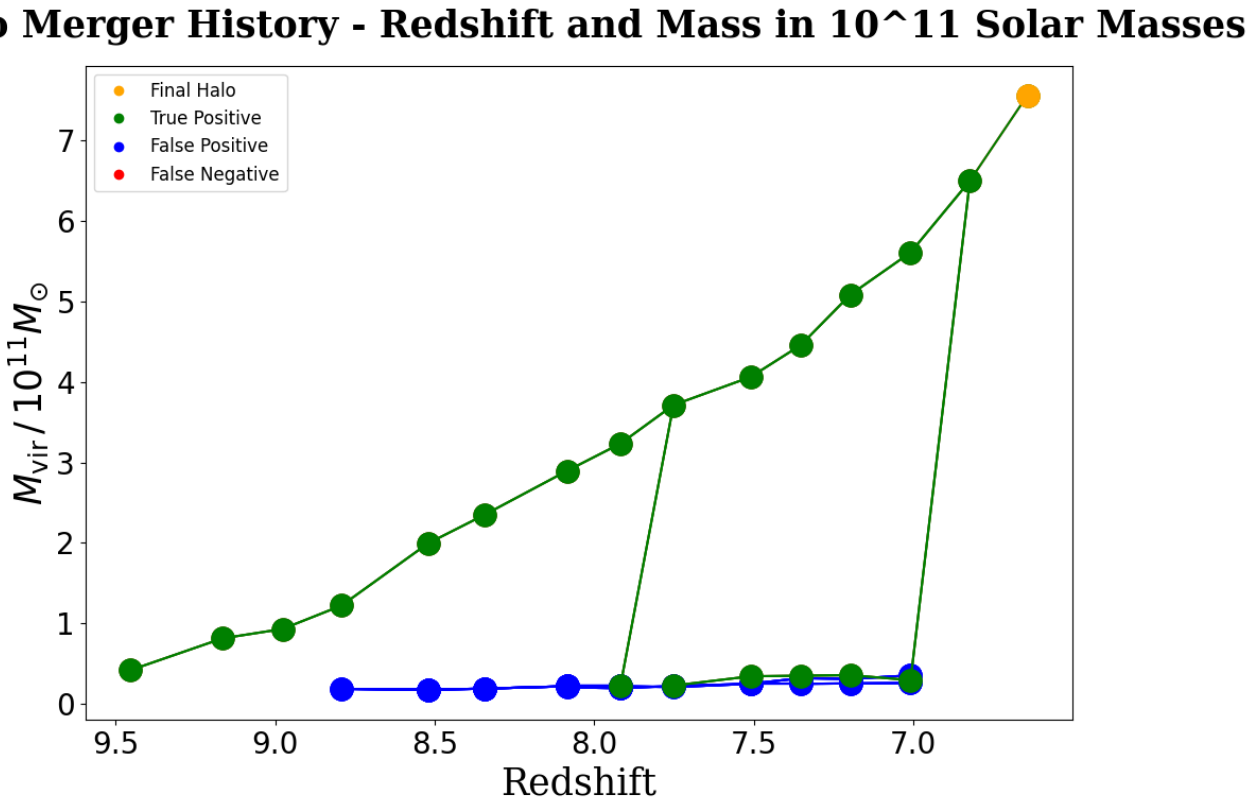

FIG. 16. Shows the comparison of the real and predicted merger tree of Halo 36000000000001 in Cluster 8. This is using the neural network model with selected features. It has successfully predicated the two merger events leading up to the target halo while having five false positives.

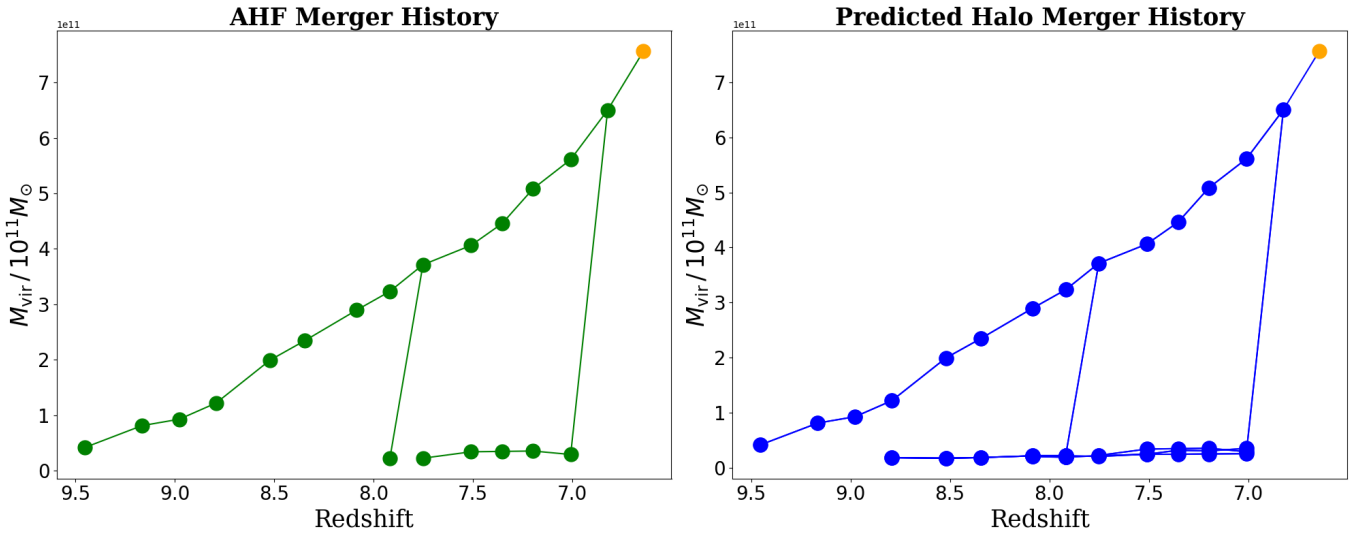

FIG. 17. Shows the real history of Halo 36000000000001 constructed from the AHF halo finder and the predicted history using neural network model with selected features.

Appendix F

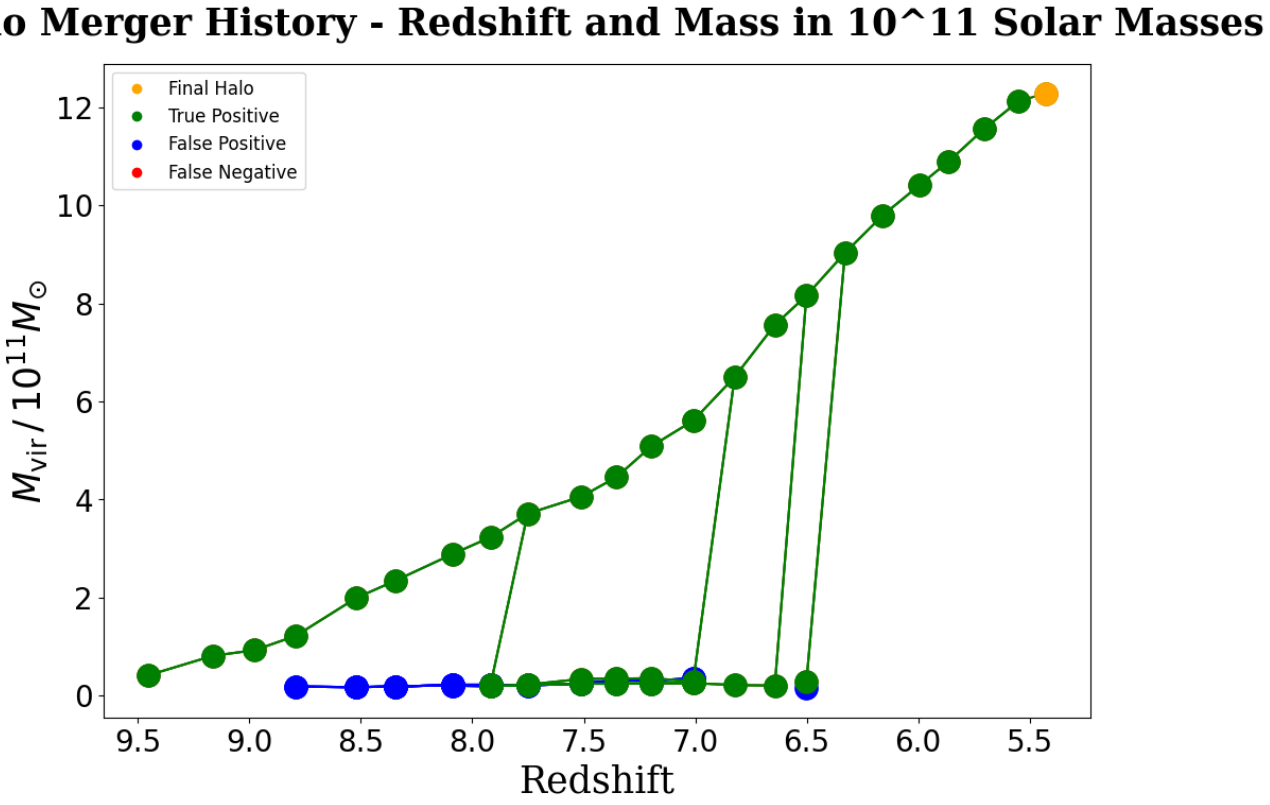

FIG. 18. Shows the comparison of the real and predicted merger tree of Halo 44000000000001 in Cluster 8. This is using the neural network model with selected features. It has successfully predicated the four merger events leading up to the target halo while having five false positives.

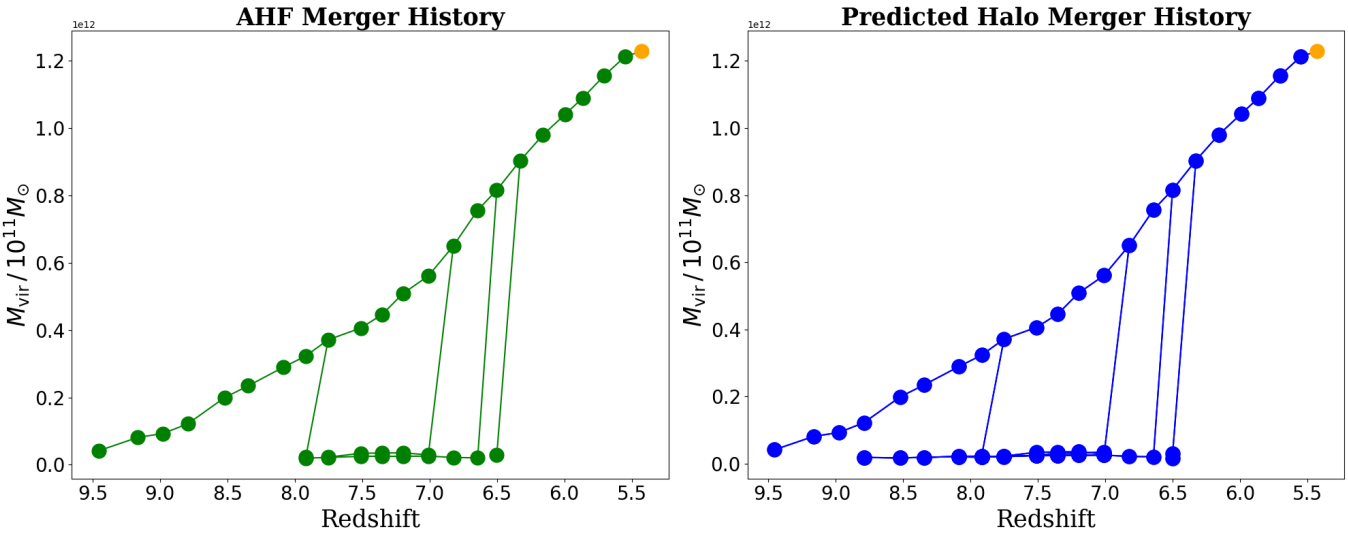

FIG. 19. Shows the real history of Halo 44000000000001 constructed from the AHF halo finder and the predicted history using neural network model with selected features.