I. PREDICTING NUMBER OF SATELLITES IN A HALO

In this section, we try to predict the number of satellites above a threshold mass, $m_{\rm above} = 1.42 \times 10^{10}\ h^{-1}\ M_\odot$, a dark matter (DM) halo can have if we are given the halo properties. The strategy to do so is as follows:

1. Train a Random Forest (RF) Regression model on the halo data. The halo data contains halo properties, or *features*, and the number of satellites, or the *ground truth*.

2. Plot feature importances, and then select the top 5-7 features.

3. Plot the correlation matrix of the features, and then select the most important uncorrelated features.

4. Run the trained RF regression model on the entire halo data set to predict number of satellites per halo.

A. Data Preparation

We begin with separating out all satellites that are above $m_{\rm above}$. We then identify the unique halos in this subset. The rest of the halos in the satellite data must have no satellites above $m_{\rm above}$. In total, there are about 1.83 million unique halos which contain the satellites in the data. Training this huge dataset is computationally very expensive. We therefore must select a small sample of the halos and then train our machine learning model on it. Sampling the halos is not straightforward. The distribution of halos per mass bin isn't uniform (Figure 1). Therefore, we must stratify the sample based on mass distribution. We created a stratified sample from 10% of the halo data set. The rest 90% of the data forms the test set.

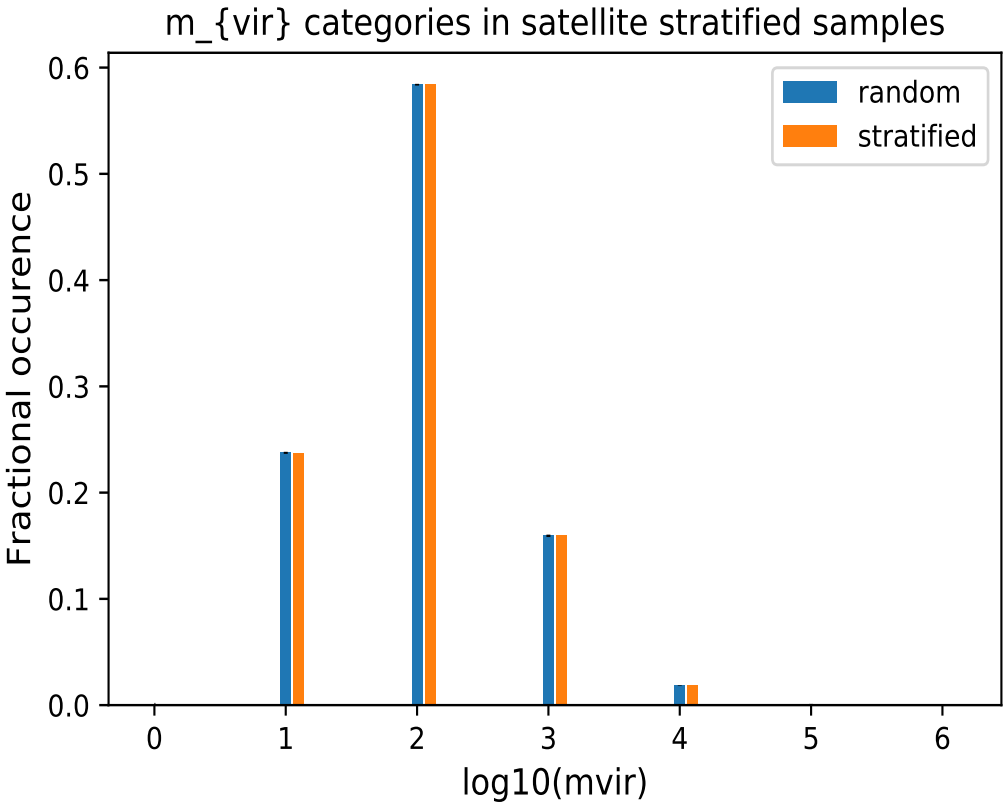

FIG. 1: Distribution of halo masses, $m_{\rm vir}$, in the halo data.

B. Random Forest Regression

We then train a RF regression model on the halo sample. The `RandomForestRegressor` module in scikit-learn has a hyper-parameter called `max_features`. We trained our sample with `max_features = log2` and `1.0`.

C. Feature Importances

Shown below are the bar plots denoting the feature importances of the RF regression model for the two hyper-parameters, `log2` and `1.0` (see Figure 2). We see that in both models, `mvir` is the most important feature. However, in `1.0`, `mvir` is more

dominant as compared to other features. In what follows below, we evaluate how good the model `1.0` is.

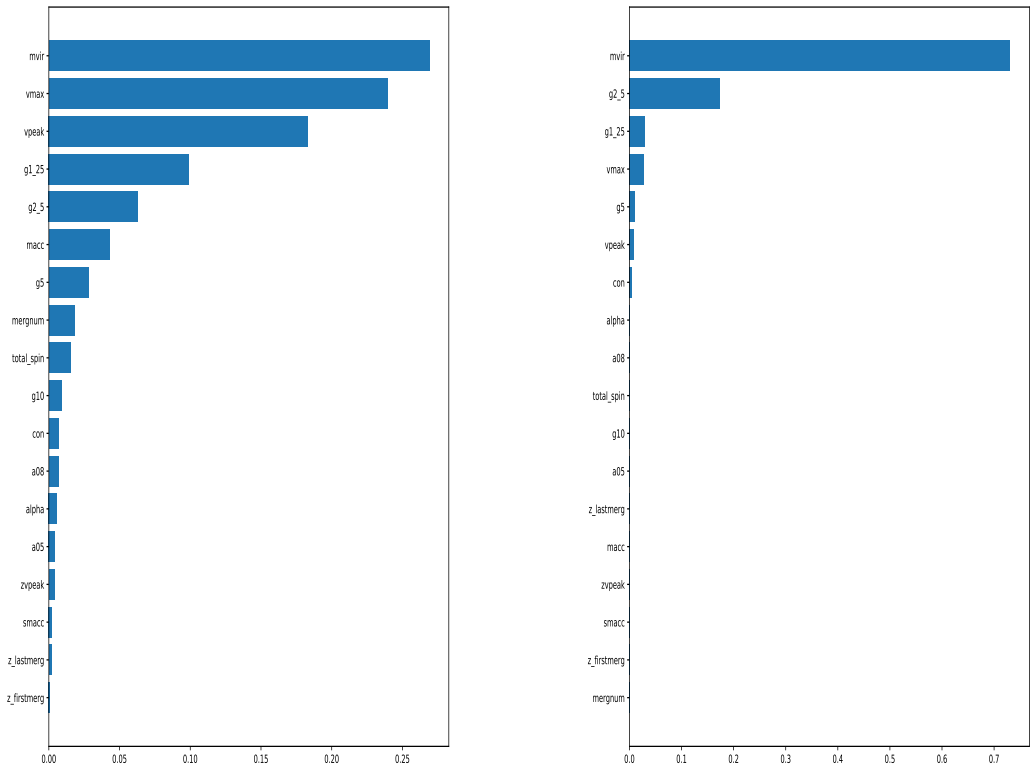

FIG. 2: Feature importances of the RF regression model for satellite galaxies. Left: `max_features = log2`, Right: `max_features = 1.0`. It appears that in `log2`, the feature importances are spread over all halo features, whereas in `1.0`, halo mass `mvir` is by far the most dominant feature followed by `g2_5, g5,` and `g1_25`.

We plot the top 7 feature correlation matrices for the two models in Figure 3. We see that in `log2`, the most important features such as `mvir, vmax` and `vpeak` are correlated to each other: `vmax-vpeak` ≈ 1.0, `vmax/vpeak-mvir` $= 0.6$. On the other hand, in `1.0`, the top 2 features, `mvir` and `g2_5`, aren't correlated to each other

(`mvir-g2_5` = 0.25). The next feature `g5` is correlated to `g2_5`, `g2_5-g5` = 0.87. However, its importance is quite low in comparison to `g2_5`. It seems like the `1.0` model is doing much better than the `log2` model since it prioritizes uncorrelated features.

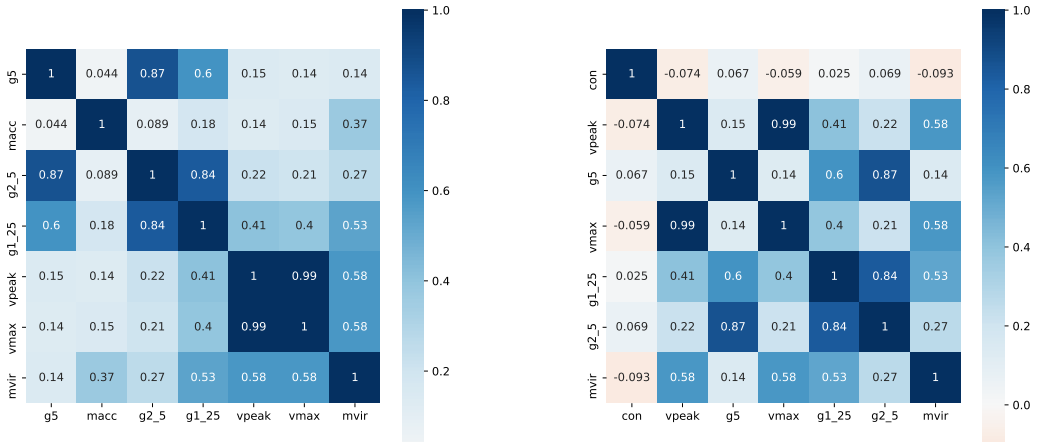

FIG. 3: Correlation matrix or heatmap of the top 7 features for the satellite data. Left: `max_features = log2`, Right: `max_features = 1.0`. The closer the values are to ± 1, the more correlated/anti-correlated the features are.

We next compare the performance measure, r^2 score, from the two models. For this step, we select the most important, uncorrelated features from both models. For `log2`, the features are `mvir, vmax, g1_25, g2_5, macc`, and for `1.0`: `mvir, g2_5, g1_25, con`. We then perform 5-fold cross-validation test on both models. After training the models, we predict the satellite numbers in the test set, followed by the entire halo data. The results are as follows, Table II:

We see that the model `1.0` does slightly better than the `log2` model. This leads to the following conclusion: the features `mvir, g2_5, g1_5, con` are good predictors of the number of satellites a halo can occupy.

Model	5-fold CV	Test	Total
`log2`	0.916 ± 0.016	0.875	0.881
`1.0`	0.922 ± 0.014	0.878	0.885
`Gradient Boost`	0.916 ± 0.017	0.887	0.895

TABLE I: Model comparison using r^2 score using both halo and environment properties.

Model	5-fold CV	Test	Total
`log2`	0.846 ± 0.047	0.866	0.867
`1.0`	0.849 ± 0.038	0.867	0.868
`Gradient Boost`	0.842 ± 0.033	0.853	0.860

TABLE II: Model comparison using r^2 score using only halo properties.

II. PREDICTING CENTRAL GALAXY OCCUPATION

The purpose of this section is to predict the occupation of the halo by a central galaxy of mass above $m_{\rm above}$. The strategy for doing so is nearly the same as that in the previous section for the case of satellites. The only difference is that central occupancy prediction is a classification problem.

A. Data Preparation

In total, the central galaxy dataset contains 8.76 million unique halos. This means that instead of using the entire dataset to train a classification model, we must create a sample from the original dataset for training. The distribution of halos per `vmax` bin isn't uniform (Figure 4). Therefore, we must stratify the sample based on `vmax` distribution. We created a stratified sample from 10% of the central data set. The rest of the 90% of the central data forms the test set.

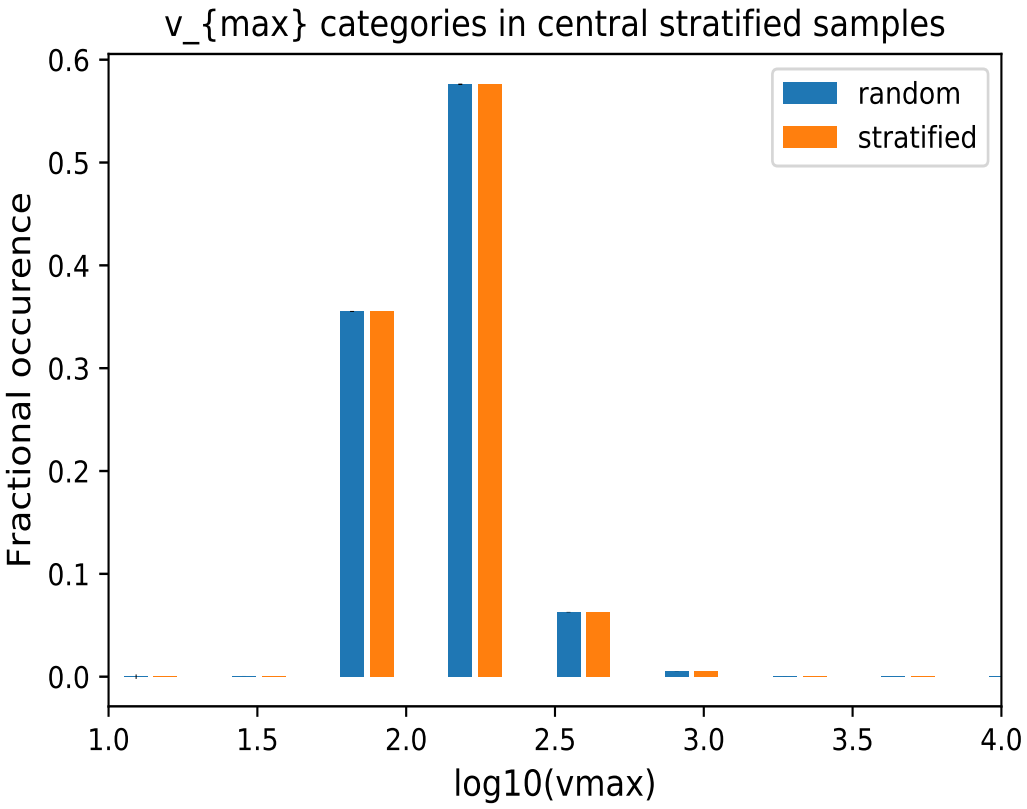

FIG. 4: Distribution of halo `vmax`, $v_{\rm max}$, in the central data.

B. RF classifier

We trained a RF classifier model on the halo sample with `max_features = log2` and `1.0`.

C. Feature Importances

Below are bar plots denoting the feature importances of the RF classifier model (Figure 5) for `max_features = log2` and `1.0`. From both models, we find that `vmax` is the most important feature. However, the `1.0` model gives `vmax` much greater importance that the `log2` model.

We then select the top 7 features from the above bar plots, and plot the heatmaps for both models (Figure 6).

Again, comparing both heatmaps, we find that the `1.0` model gives more importance to uncorrelated features. We then train our samples with both models keeping only the most important uncorrelated features. For `log2`, we select `vmax, mvir, total_spin, z_firstmerg, macc`, and for `1.0`, we select `vmax, z_lastmerg, a05, total_spin`. We plot the confusion matrices from the validation set (25% instances from the stratified sample) from both models below (Figure 7):

Finally, we compare the various F_1 scores from both models as shown in the table below, Table III

Model	5-fold CV	Test	Total
`log2`	0.904 ± 0.005	0.904	0.905
`1.0`	0.905 ± 0.002	0.905	0.905

TABLE III: Model comparison using F_1 score.

We see that, the `1.0` model performs equally well as the `log2` model.

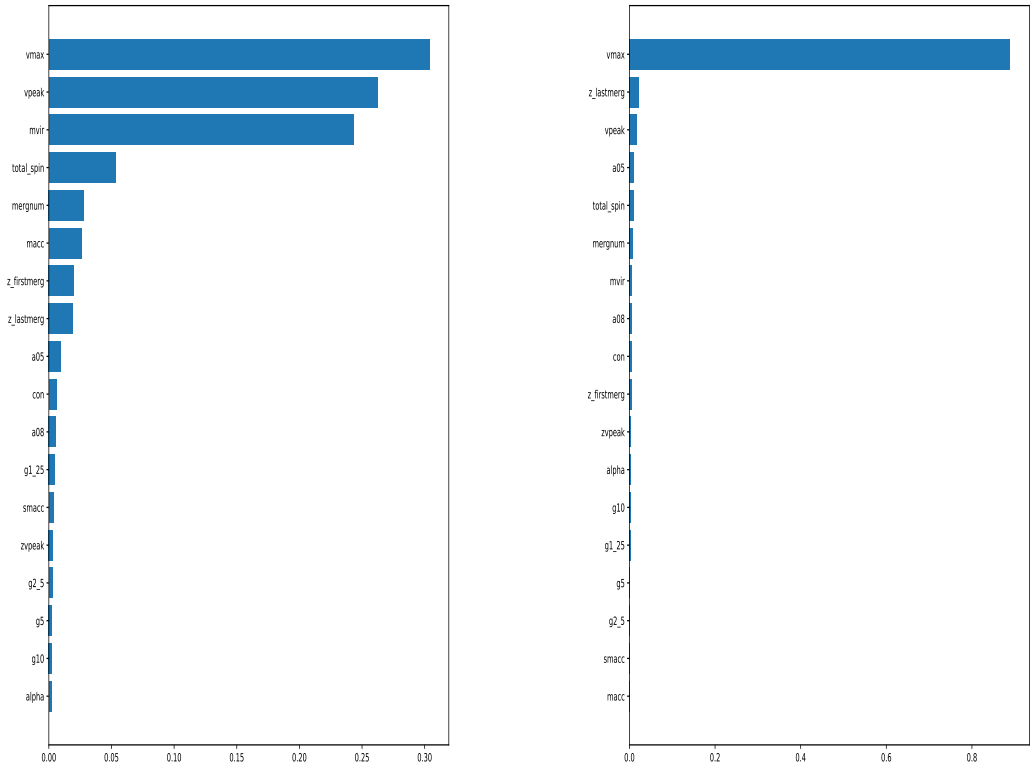

FIG. 5: Feature importances of the RF classifier model for central galaxies. Left: `max_features = log2`, Right: `max_features = 1.0`. It appears that in `log2`, the feature importances are spread over all halo features, whereas in `1.0`, halo mass `vmax` is by far the most dominant feature followed by `z_lastmerg, vpeak,` and `a05`.

III. PREDICTING BOTH CENTRAL AND SATELLITES

In this section, we predict the total number of galaxies, central and satellites, in a given halo. We can do this in 2 different ways: first, we can combine the predictions of the regressor and classifier as described above. Second, we can run a regressor predicting both central and satellites.

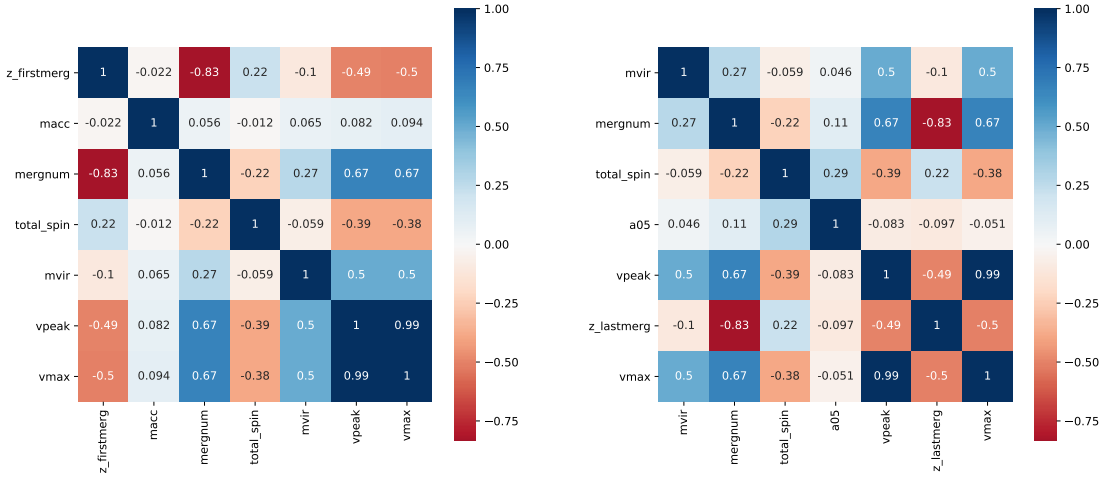

FIG. 6: Correlation matrix or heatmap of the top 7 features for the central data. Left: `max_features = log2`, Right: `max_features = 1.0`.

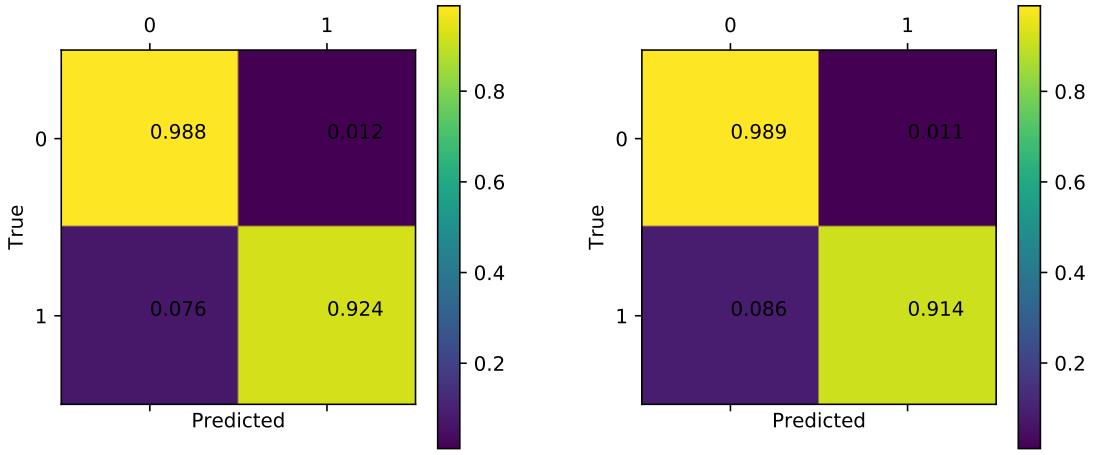

FIG. 7: Confusion matrix of the validation set for uncorrelated important features for central galaxies. Left: `max_features = log2`, Right: `max_features = 1.0`.

We can compare both methods by plotting the number of galaxies per halo mass

G	5-fold CV	Test	Total	Top 4 + env	All
0.0	0.905 ± 0.003	0.905	0.905	0.907	0.922
0.1	0.891 ± 0.003	0.891	0.891	0.891	0.922

TABLE IV: Model comparison using F_1 score with `max_features` = 1.0.

Features	Total (G = 0)	Total (G = 0.1)
`Top 4`	0.915	0.919
`Top 4 + env`	0.919	
`All`	0.930	–

TABLE V: Model comparison using F_2 score.

for each individual halo. This is shown in Figure 8. A key feature which stands out for both models is that they fail spectacularly at high halo mass regimes, $m_{\rm halo} > 10^4$. This seems to be a result of low number of halo samples at higher halo masses.

We then plot the galaxy occupation per total number of halos in a given mass bin or the halo occupation function as a function of halo mass, Figure 10. At low masses, $\log_{10}(m_{\rm halo}) < 4$, the first method does better. This is so because the RF regressor does a poor job predicting small number of galaxies in a given halo.

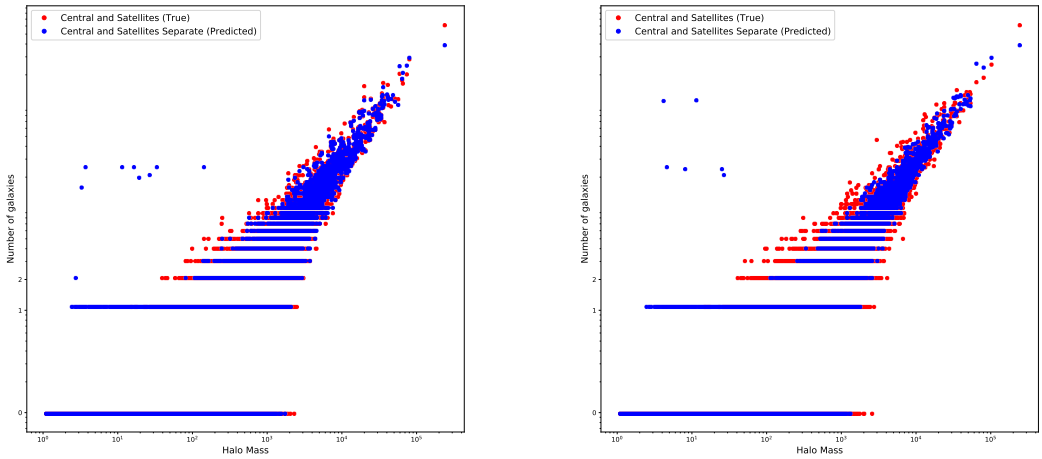

FIG. 8: Number of galaxies vs. halo mass. The halo masses shown are in units of $10^{10}\ h^{-1}\ \mathrm{M}_{\odot}$. Left: Plot for the training set used to train the gradient boost regressor model. Right: Plot for a randomly selected sample.

Appendix A: R^2 score or coefficient of determination

Let $\bar{y}$ denote the mean of the ground truth values. The total sum of squares is given as

$$SS_{tot} = \sum_i (y_i - \bar{y})^2. \tag{A1}$$

The regression/explained sum of squares is given as

$$SS_{reg} = \sum_i (p_i - \bar{y})^2 \tag{A2}$$

where p_i are the predicted values. The residual sum of squares is given as

$$SS_{res} = \sum_i (p_i - y_i)^2 = \sum_i e_i^2. \tag{A3}$$

The R^2 is given as

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}. \tag{A4}$$

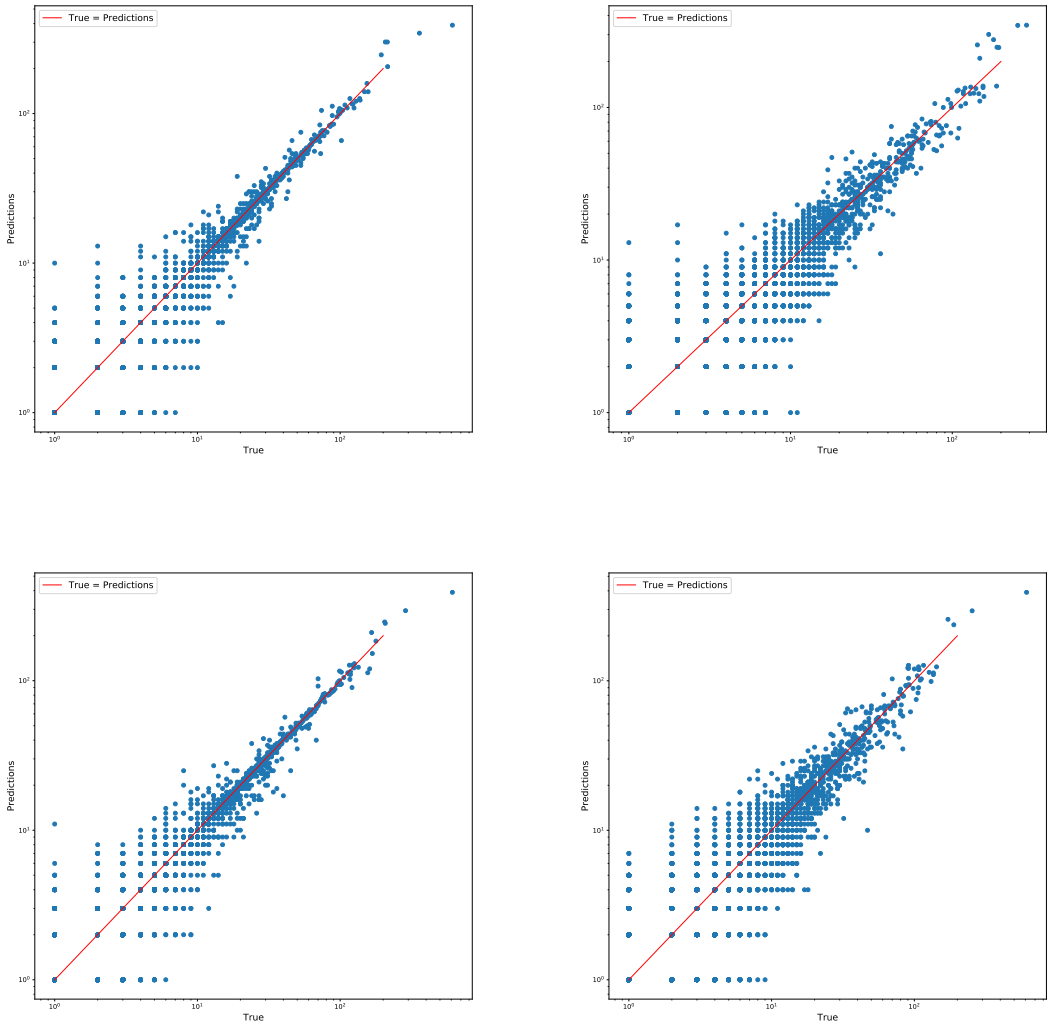

FIG. 9: Prediction vs. true values for the training set (left) and a randomly selected sample (right). There seems to be less scatter on the figure on the left since this represents the training set.

The best model where $p_i = y_i$ will give an R^2 value of 1 whereas the worst model will have an R^2 value of $-\infty$.

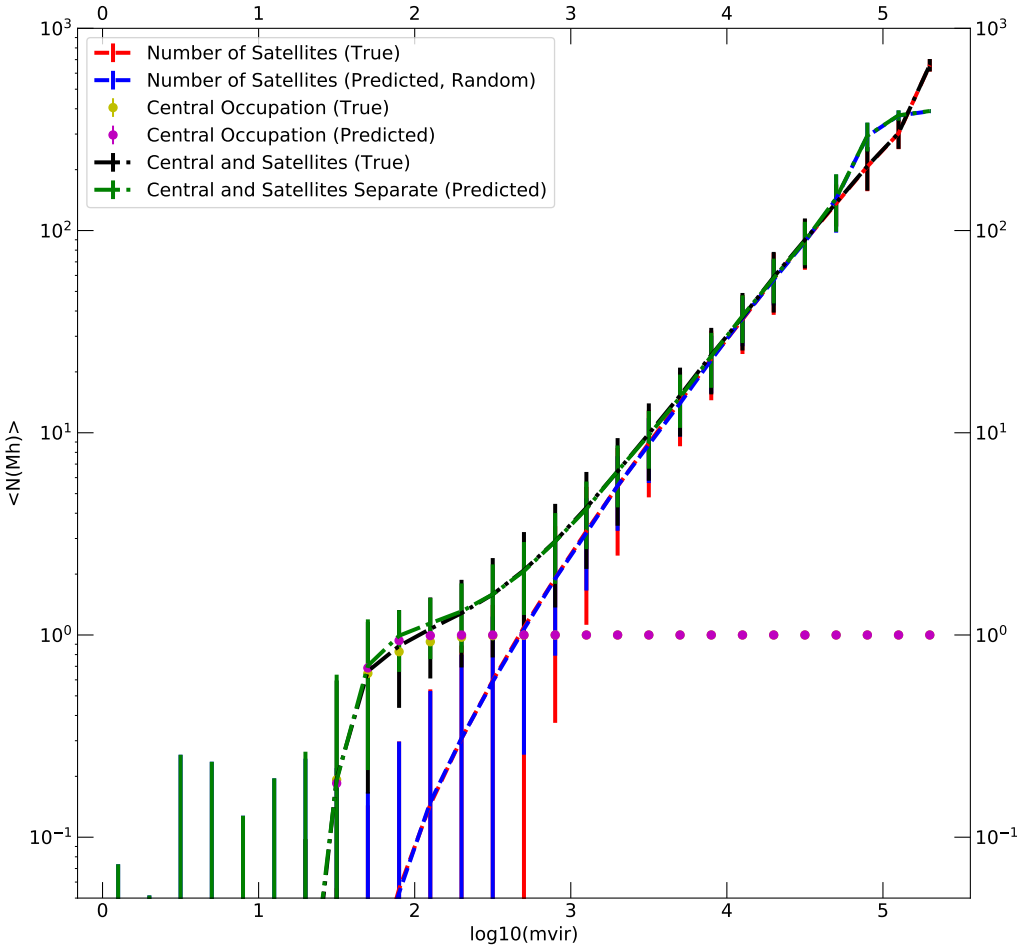

FIG. 10: Halo occupation vs. log10 of halo mass.

Appendix B: Heatmap scores

The scores given in the heatmap cells are the Pearson's r given as

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}. \tag{B1}$$

Therefore the r score can only take values between -1 and 1.

Appendix C: Decision Tree: Regression

Below is an example of a decision tree for a regressor. How does a decision tree

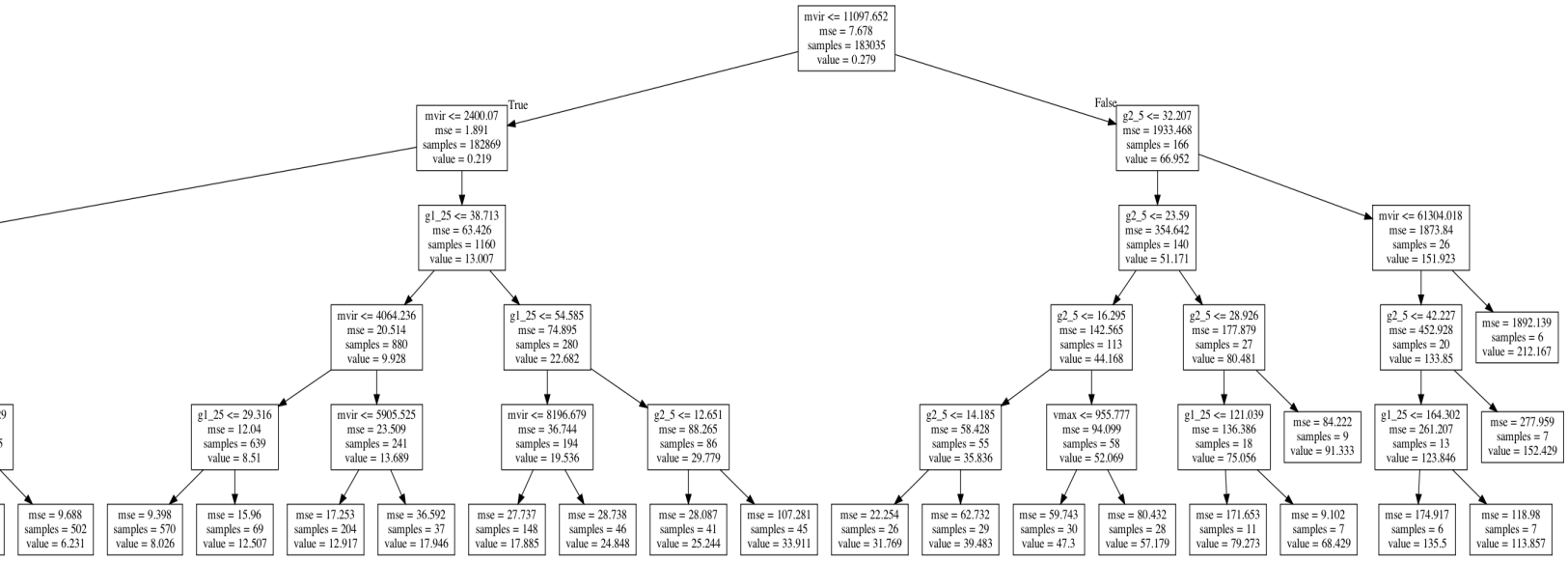

FIG. 11: Example of a decision tree with `max_depth` = 5 and `max_features` = 1.

make decisions? The CART (Classification and Regression Tree) algorithm to train decision trees follows the following prescription:

1. When `max_features = 1`, at each node the tree has all features available from which it picks the best feature to split the data.

2. The best feature, k, and the threshold, t_k, is selected by minimizing the loss function $J(k, t_k)$. Eg., $k =$ `mvir`, $t_k = 2400.07$ in the first daughter node on the left. The loss function is given by

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right} \tag{C1}$$

where

$$MSE_{node} = \sum_i (p_{node} - y_i)^2 \tag{C2}$$

$$p_{node} = \frac{1}{m_{node}} \sum_i y_i$$

3. In each node, `mse` denotes the mean square error, `samples` denotes the number of samples in the node/leaf, and `value` denotes the predicted value of those samples which is equal to the mean value of the ground truth of the samples y_i.

Appendix D: Decision Tree: Classification

Below is an example of a decision tree for a classifier. The algorithm for making

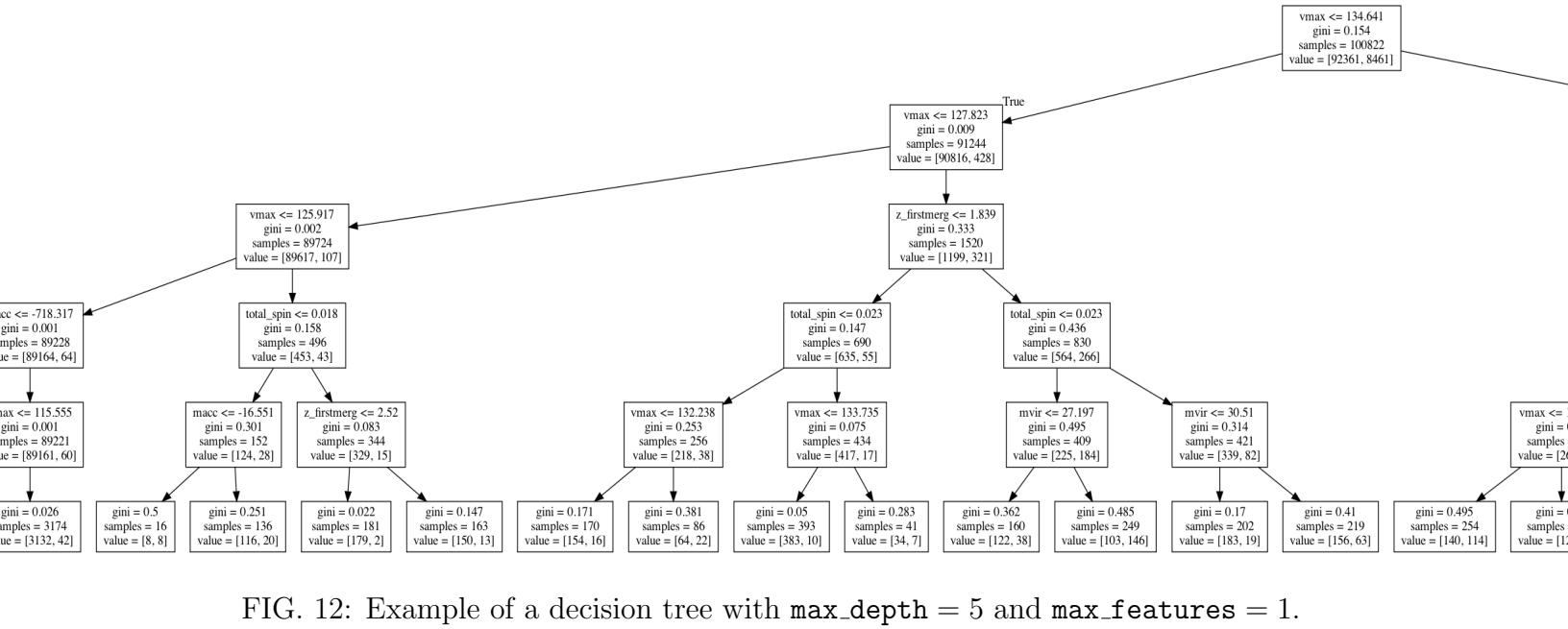

FIG. 12: Example of a decision tree with `max_depth` = 5 and `max_features` = 1.

decisions is the same as that of the regressor. The only difference is that instead of minimizing the mean square error, the classifier minimizes either the gini impurity

(default) or the entropy which are defined as

$$\text{Gini Impurity: } G = 1 - \sum_{k=1}^{n} p_k^2, \tag{D1}$$

$$\text{Entropy: } H = - \sum_{k=1,\ p_k \neq 0}^{n} p_k \ log_2(p_k) \tag{D2}$$

where p_k is the fraction of the instances belonging class k in the node. For example, in the first daughter node on the left, $p_0 = 90816/91244 = 0.995$, and $p_1 = 0.005$ giving us $G = 0.009$. In the nodes/leaves in the picture, `value` denotes the number of instances in the `samples` in that node which belong to class 0 and 1.