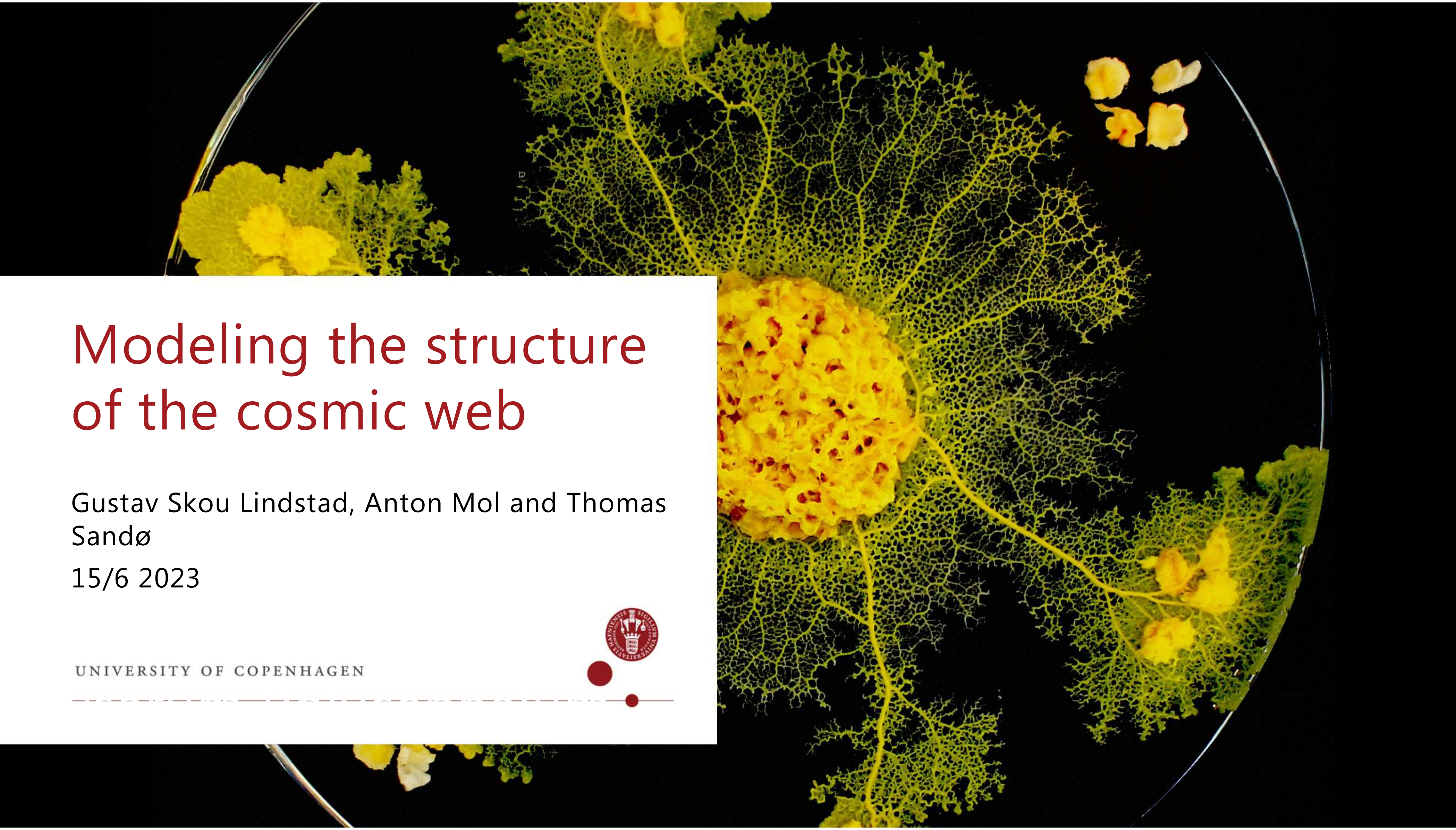


Modeling the structure of the cosmic web

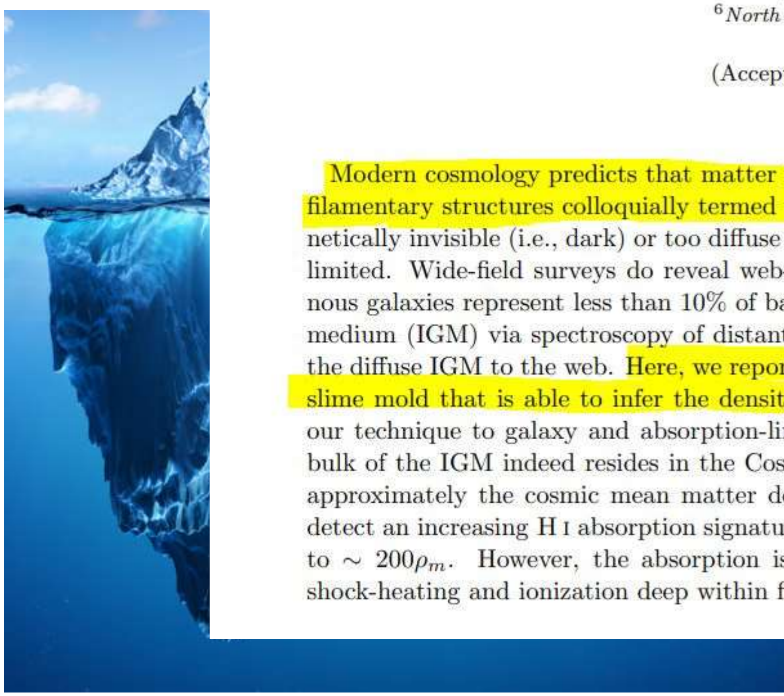
Gustav Skou Lindstad, Anton Mol and Thomas Sandø

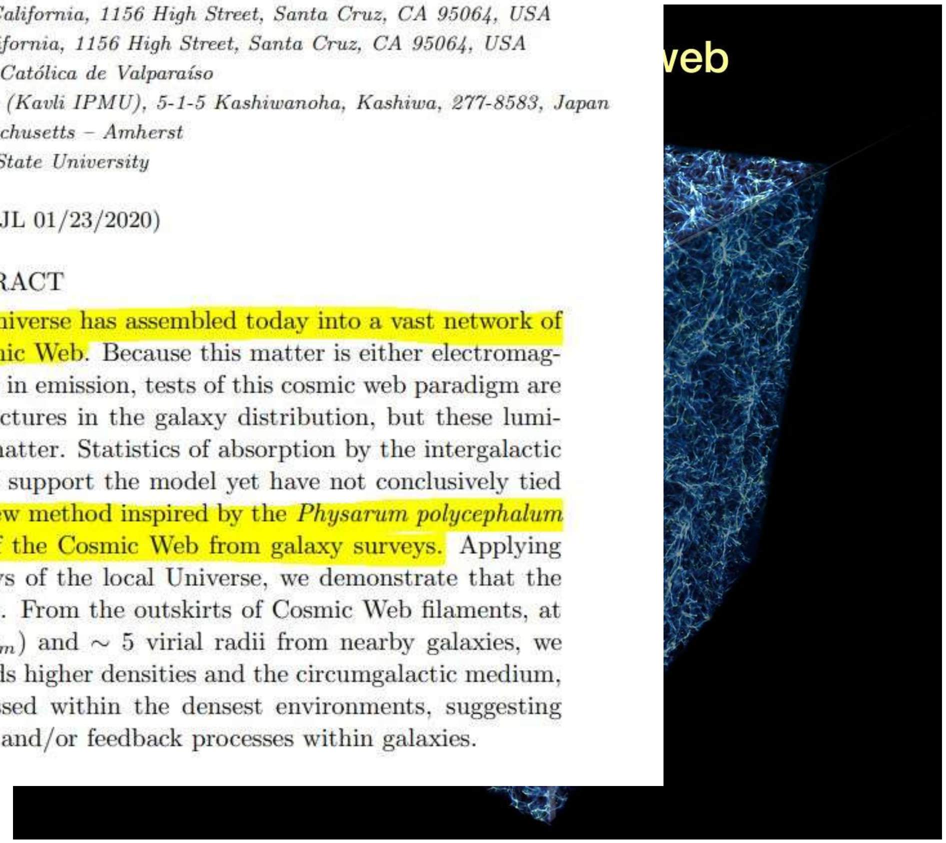
15/6 2023

UNIVERSITY OF COPENHAGEN

Motivati

- >80% see





Revealing the Dark Threads of the Cosmic Web

Joseph N. Burchett,[1] Oskar Elek,[2] Nicolas Tejos,[3] J. Xavier Prochaska,[1,4] Todd M. Tripp,[5] Rongmon Bordoloi,[6] and Angus G. Forbes[2]

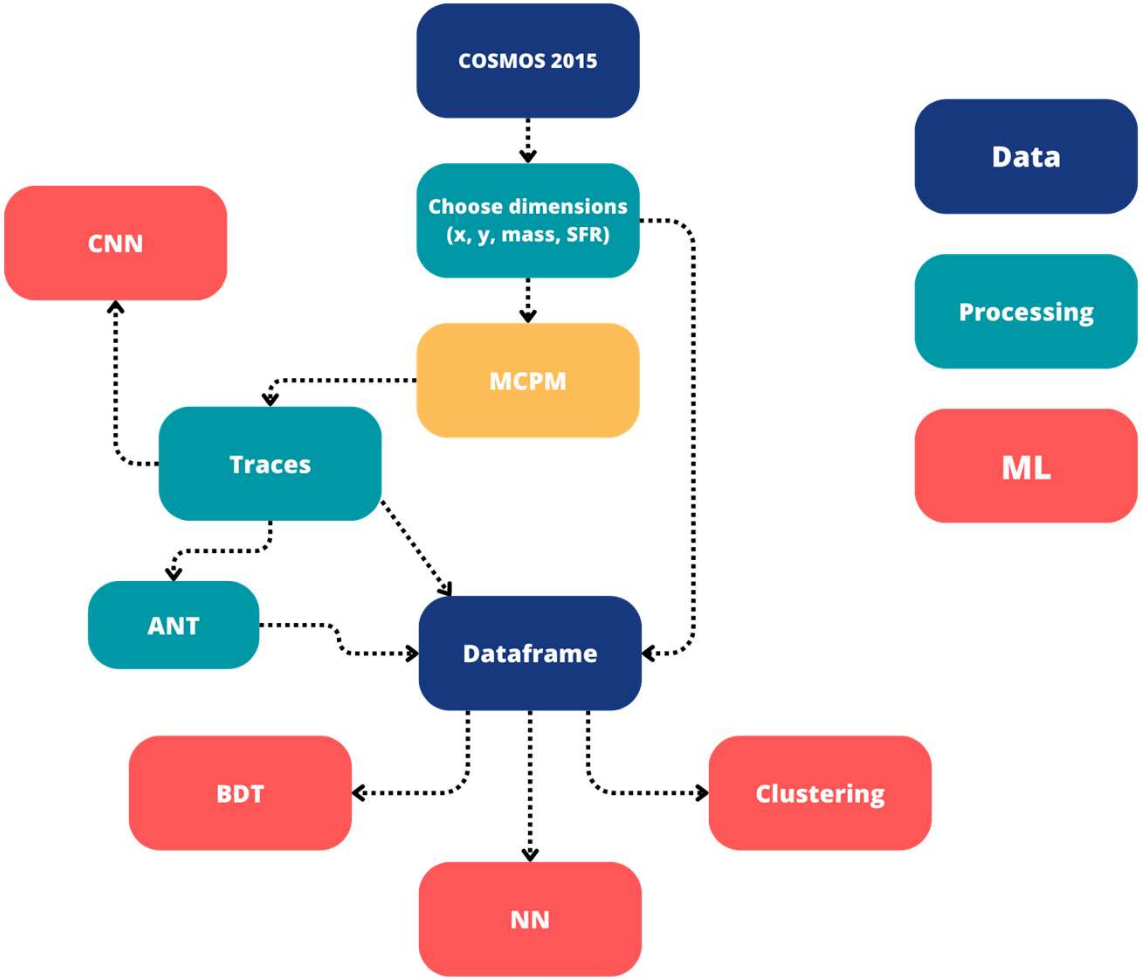
[1] Department of Astronomy & Astrophysics, University of California, 1156 High Street, Santa Cruz, CA 95064, USA
[2] Department of Computational Media, University of California, 1156 High Street, Santa Cruz, CA 95064, USA
[3] Pontificia Universidad Católica de Valparaíso
[4] Kavli Institute for the Physics and Mathematics of the Universe (Kavli IPMU), 5-1-5 Kashiwanoha, Kashiwa, 277-8583, Japan
[5] University of Massachusetts – Amherst
[6] North Carolina State University

(Accepted to ApJL 01/23/2020)

ABSTRACT

Modern cosmology predicts that matter in our Universe has assembled today into a vast network of filamentary structures colloquially termed the Cosmic Web. Because this matter is either electromagnetically invisible (i.e., dark) or too diffuse to image in emission, tests of this cosmic web paradigm are limited. Wide-field surveys do reveal web-like structures in the galaxy distribution, but these luminous galaxies represent less than 10% of baryonic matter. Statistics of absorption by the intergalactic medium (IGM) via spectroscopy of distant quasars support the model yet have not conclusively tied the diffuse IGM to the web. Here, we report on a new method inspired by the *Physarum polycephalum* slime mold that is able to infer the density field of the Cosmic Web from galaxy surveys. Applying our technique to galaxy and absorption-line surveys of the local Universe, we demonstrate that the bulk of the IGM indeed resides in the Cosmic Web. From the outskirts of Cosmic Web filaments, at approximately the cosmic mean matter density (ρ_m) and ~ 5 virial radii from nearby galaxies, we detect an increasing H I absorption signature towards higher densities and the circumgalactic medium, to $\sim 200\rho_m$. However, the absorption is suppressed within the densest environments, suggesting shock-heating and ionization deep within filaments and/or feedback processes within galaxies.

Overview

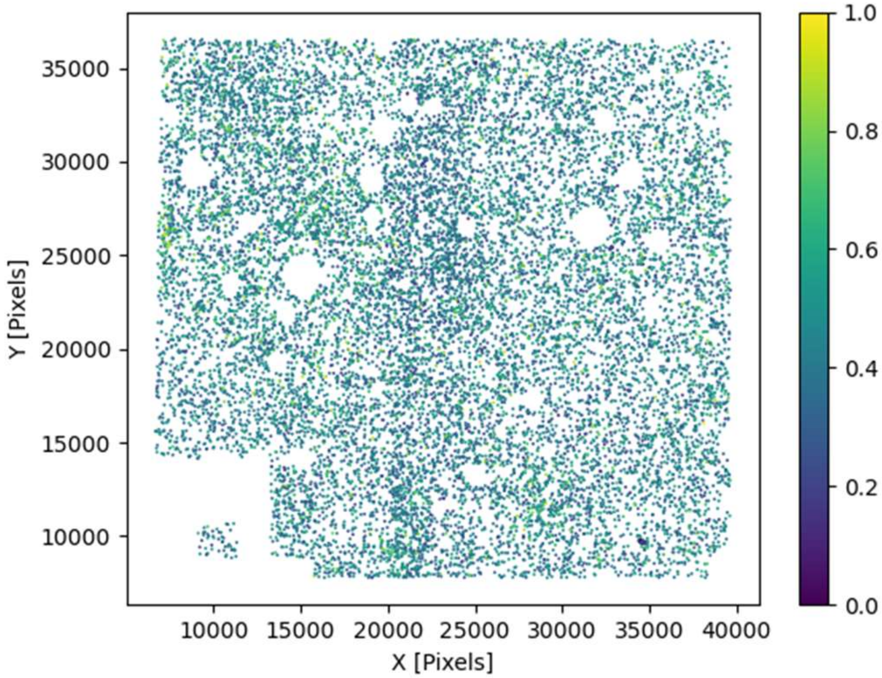

COSMOS 2015
Choose dimensions (x, y, mass, SFR)
CNN
MCPM
Traces
ANT
Dataframe
BDT
Clustering
NN
Data
Processing
ML

The Data

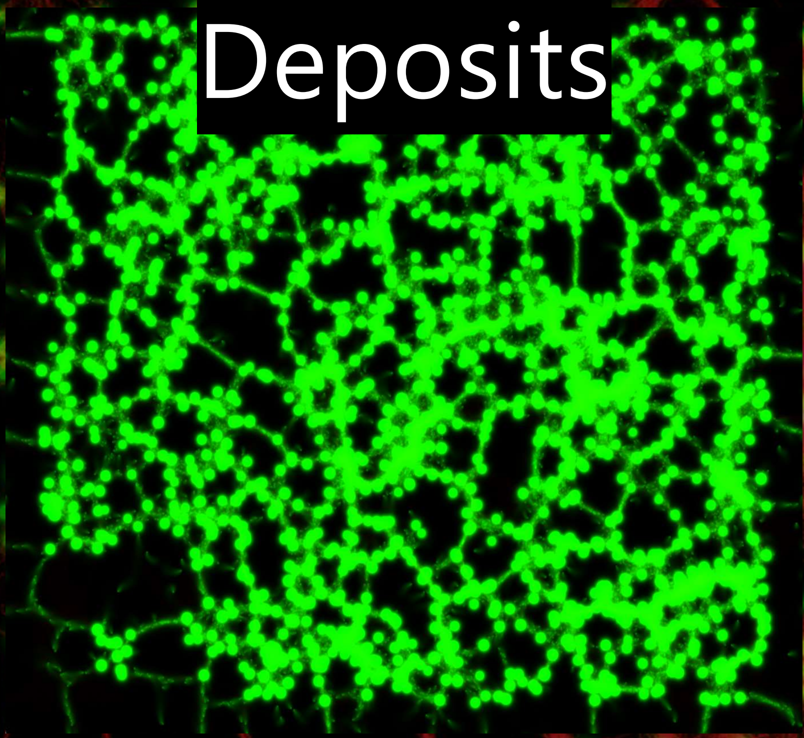
COSMOS 2015 Catalogue

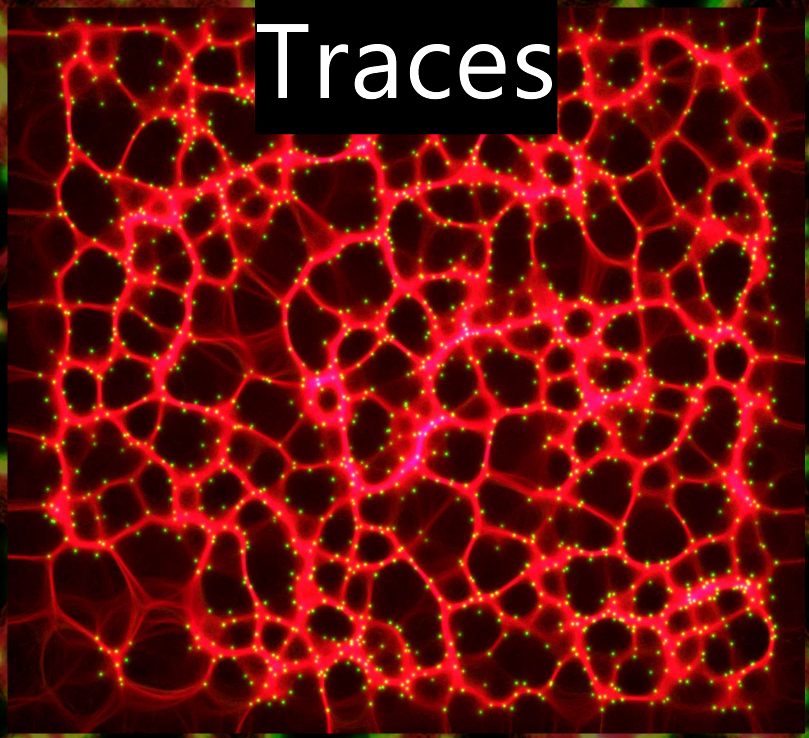
- Physical properties and location for >500,000 galaxies

- RA, DEC, X, Y, Mass, SFR, Redshift and the light in different filters



MCPM MODEL

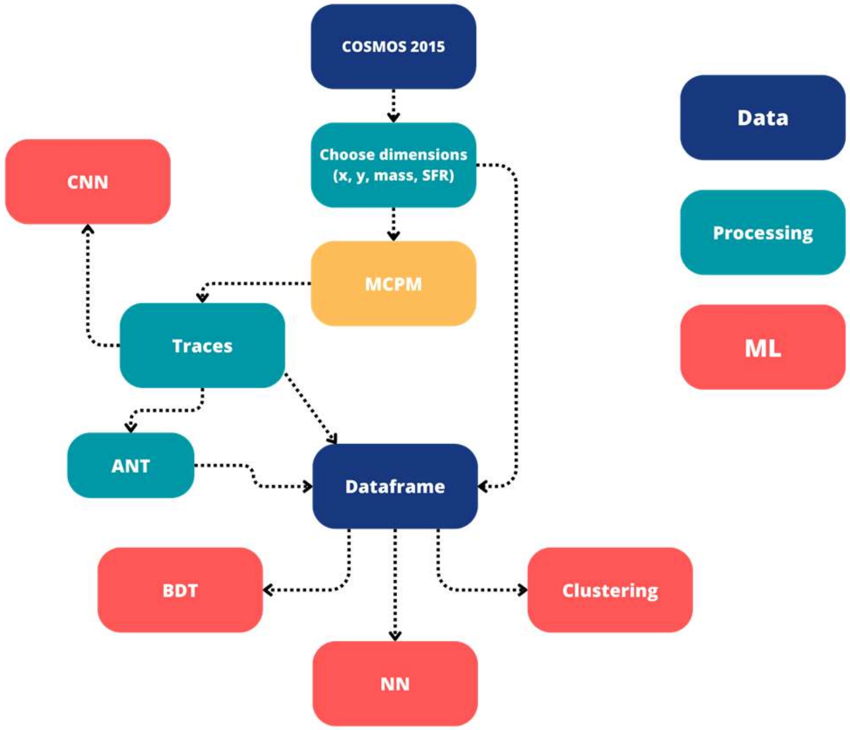




Applying Machine Learning to Trace (3 Ideas)

- Determining the mass of objects from Trace
  - Boosted Decision Tree (BDT)
  - Neural Network

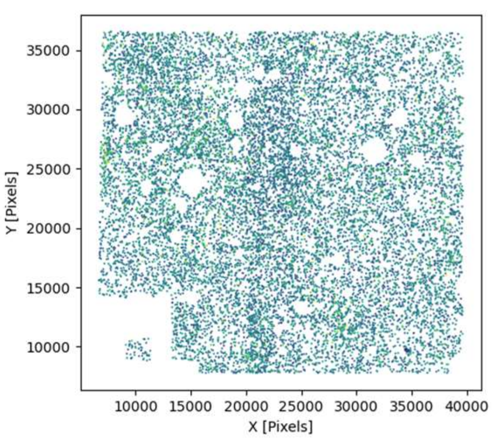
- Determining the redshift of an image from the Trace
  - Convoluted Neural Network (CNN)
  - Clustering



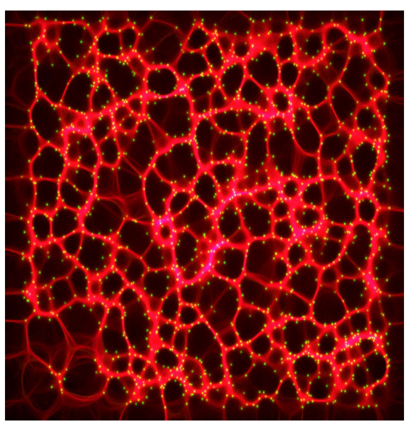
Determining the mass of galaxies – Preprocessing data

- Make data same shape
  - Trace format (1400x1364)
  - Data format (X, Y, Mass, ..
- Convert coordinates into pixels
  - In principle some information loss

Data

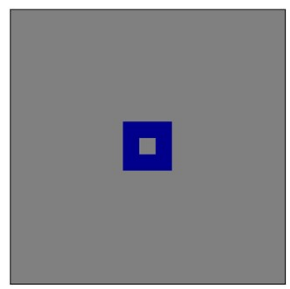


Trace

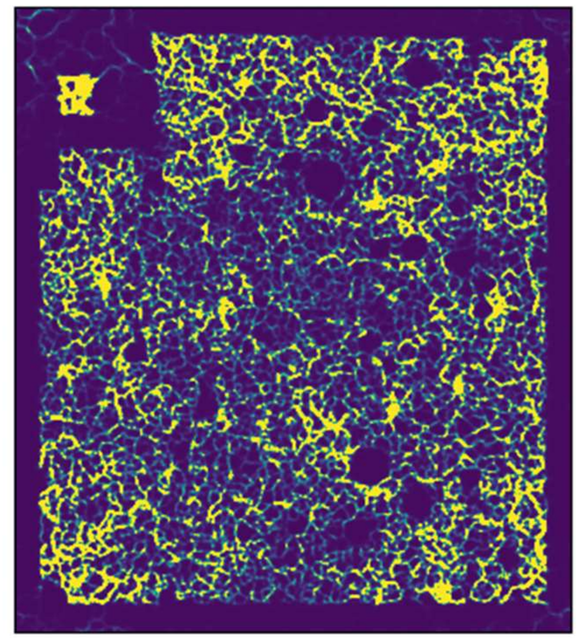


- Average of Nearby Trace (ANT)
  - 2DConvolve
  - Square kernel (Up to 17 x 17)

The Kernel



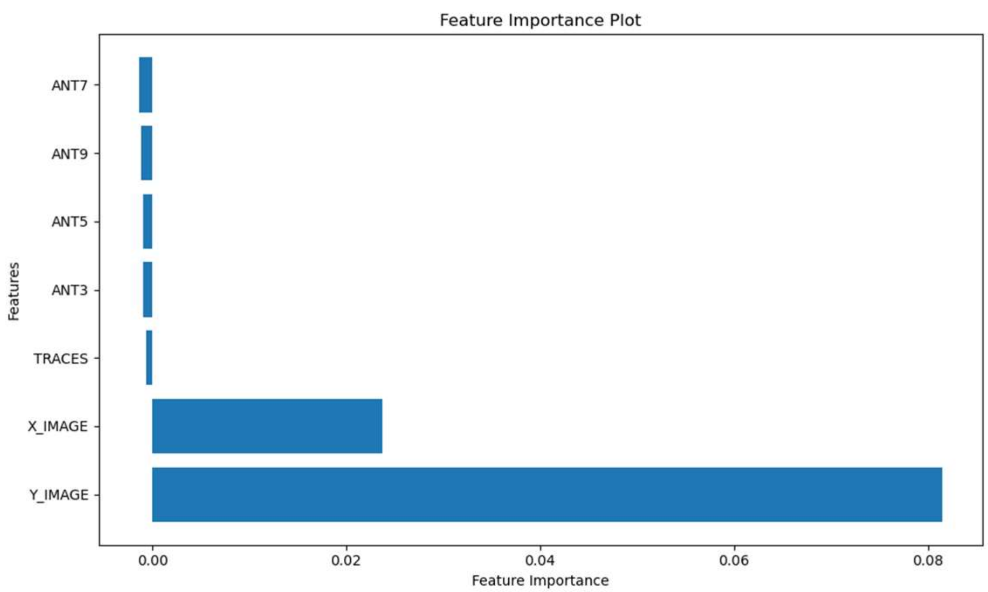
ANT



Determining the mass of galaxies – Neural Network
- Feature importance

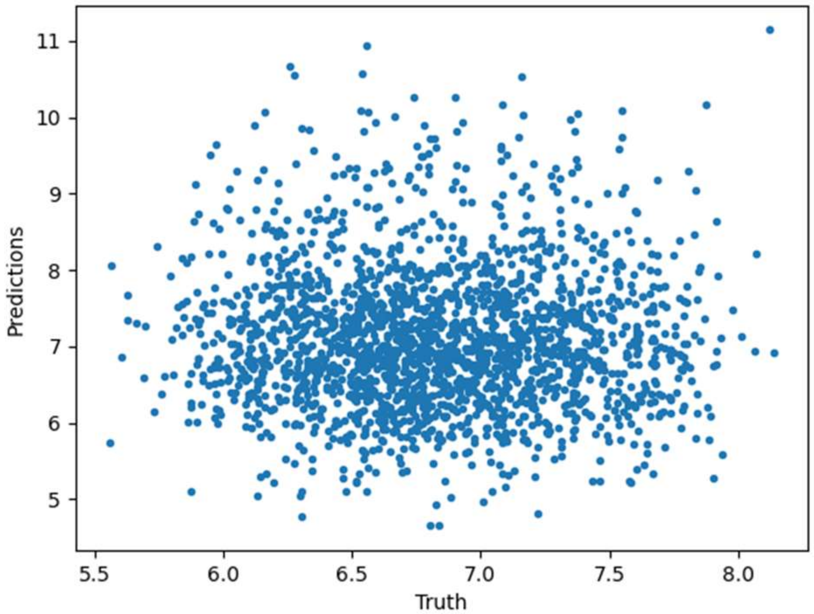
```
variable_names = ['X_IMAGE', 'Y_IMAGE', 'TRACES', 'ANT3', 'ANT5', 'ANT7', 'ANT9']
input_data     = df[variable_names]
truth_data     = df['MASS']
```

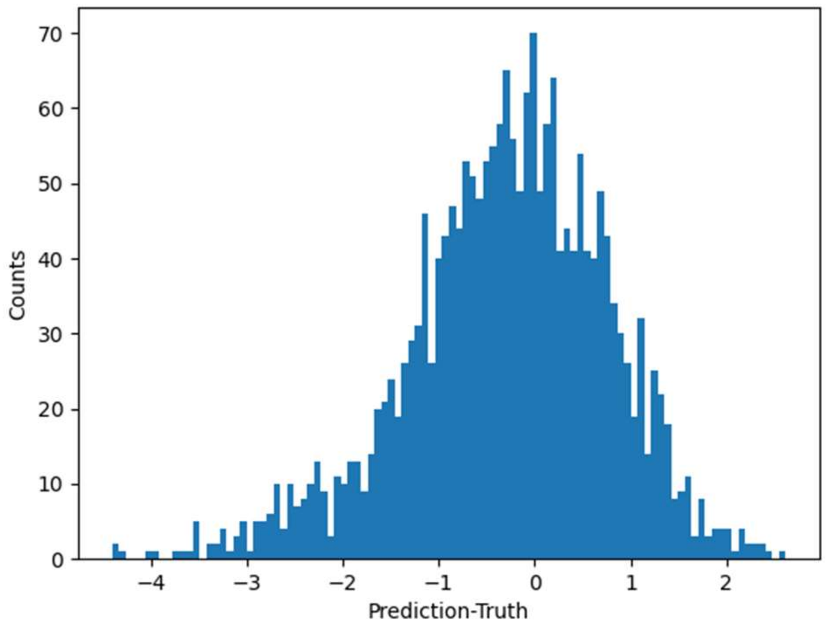
Data points: 16303
Redshift: 0.001 – 0.2
Bayesian Optimization
Best hyperparameters: {
'batch_size': 24.0,
'learning_rate': 0.00893480474068008,
'num_epochs': 38.0,
'num_layers': 1,
'num_neurons': 109.0
}



Determining the mass of galaxies – Neural Network
- Results

MSE: 1.198

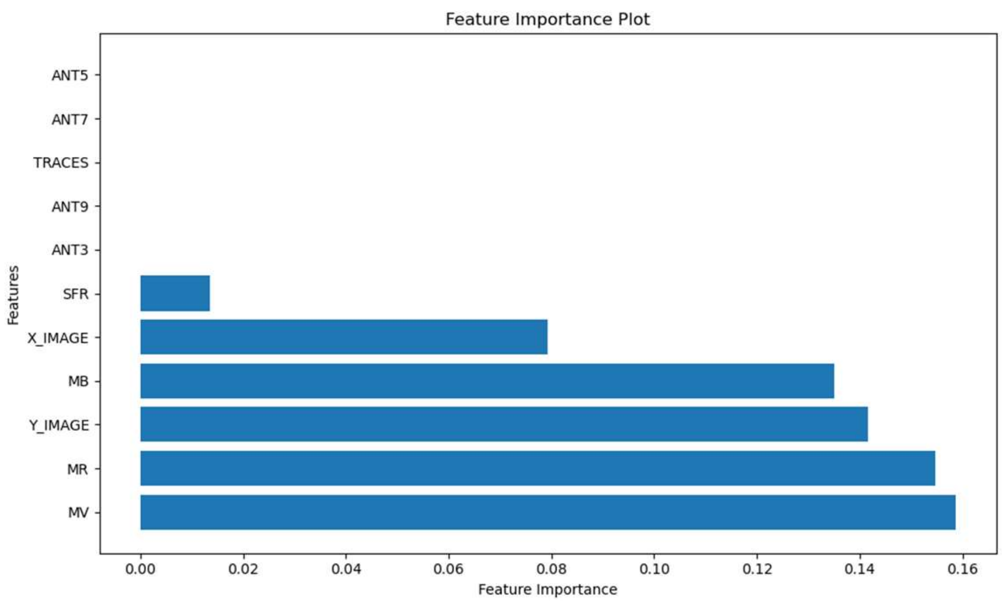




Determining the mass of galaxies – Neural Network
- Feature importance now with light

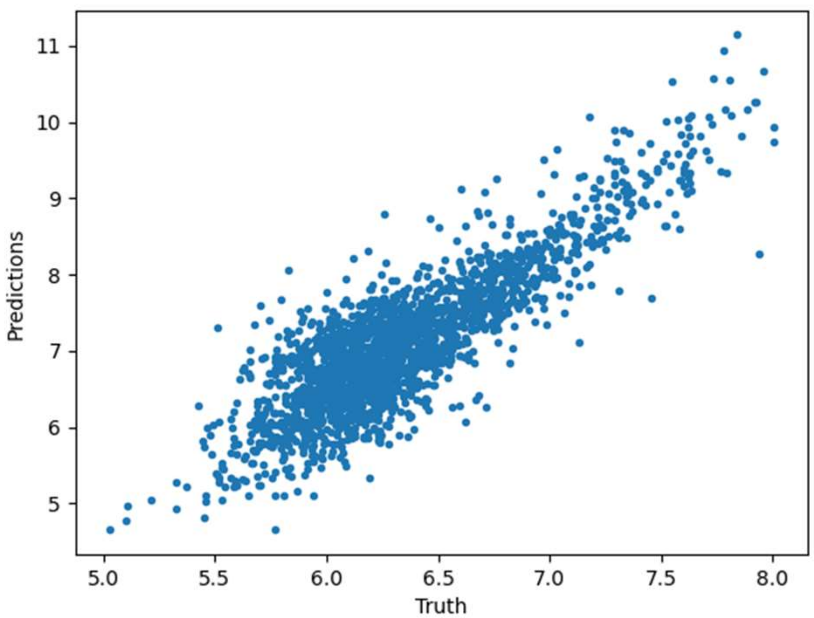
```
variable_names = ['X_IMAGE', 'Y_IMAGE', 'TRACES', 'ANT3', 'ANT5', 'ANT7', 'ANT9', 'SFR', 'MB', 'MV', 'MR']
input_data     = df[variable_names]
truth_data     = df['MASS']
```

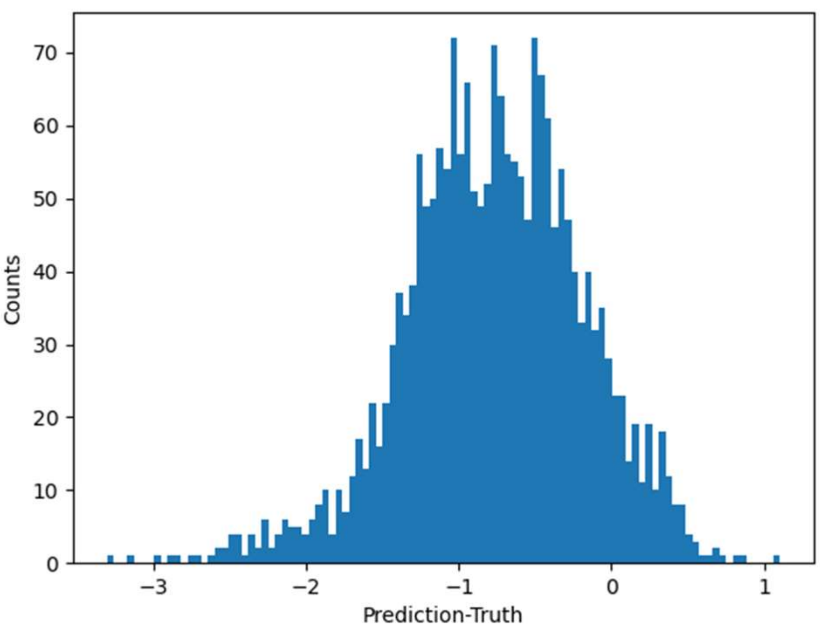
Data points: 16303
Redshift: 0.001 – 0.2
Bayesian Optimization
Best hyperparameters: {
'batch_size': 24.0,
'learning_rate': 0.00831077138618366, 'num_epochs': 49.0,
'num_layers': 1,
'num_neurons': 95.0
}



Determining the mass of galaxies – Neural Network
- Results with light

MSE: 0.570

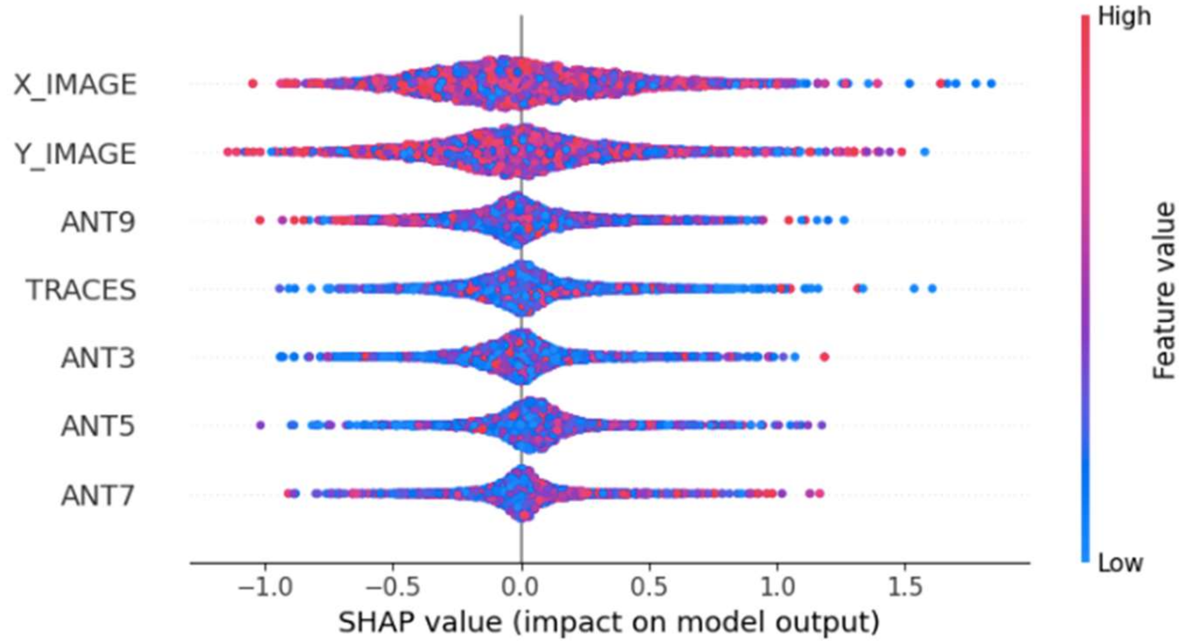




Determining the mass of galaxies – Boosted Decision Tree

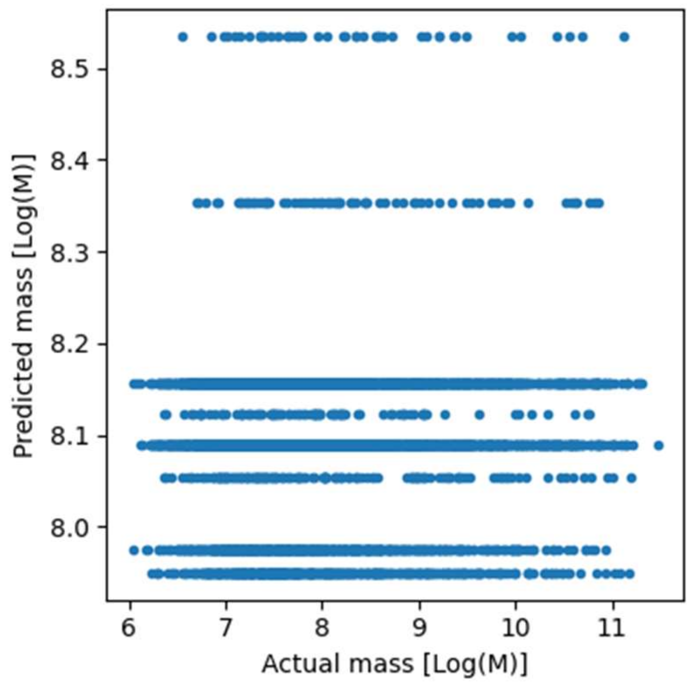
- Bayesian HP optimization:
  - max_depth: 3
  - min_samples_leaf: 100
  - min_samples_split: 94

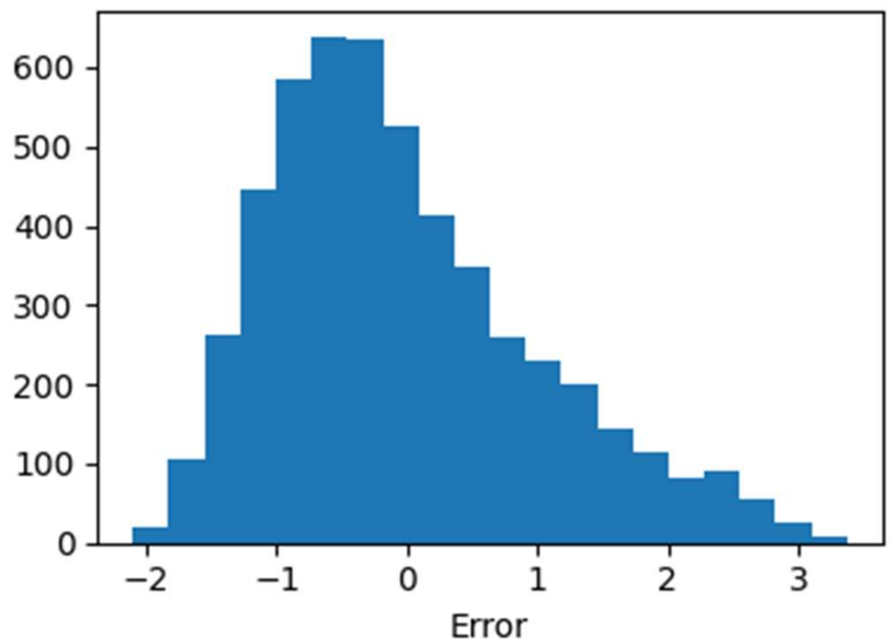
SHAP Values



Determining the mass of galaxies – Boosted Decision Tree

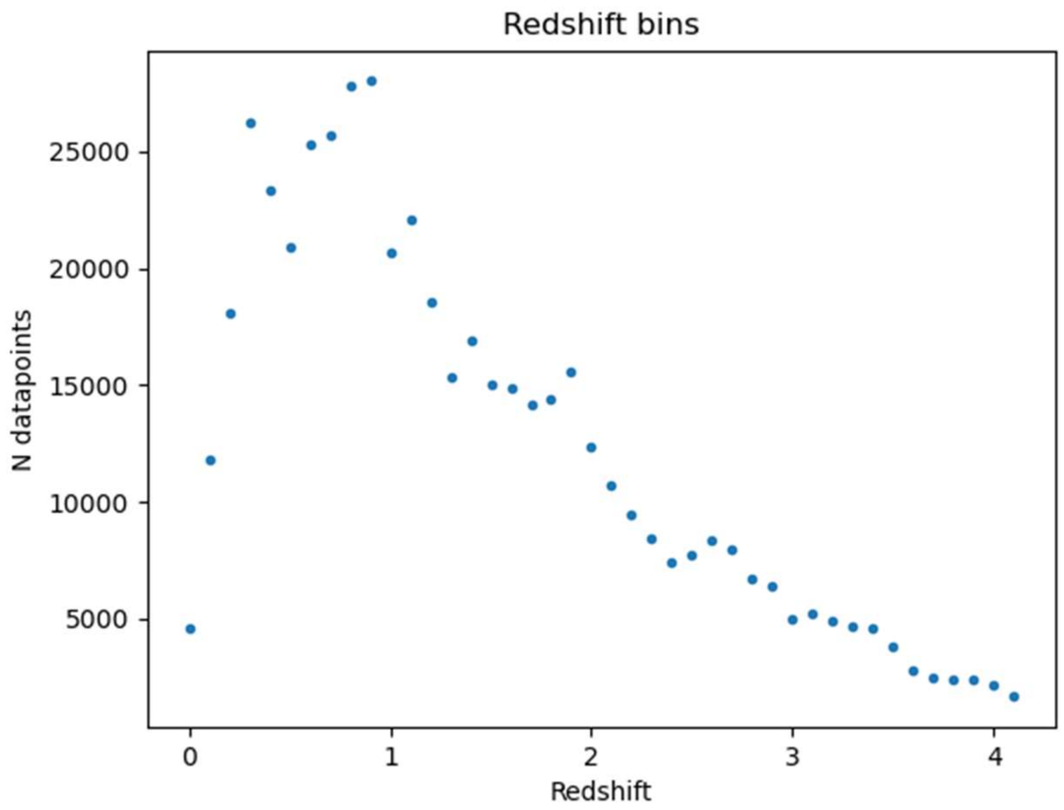
- MSE: 1.047



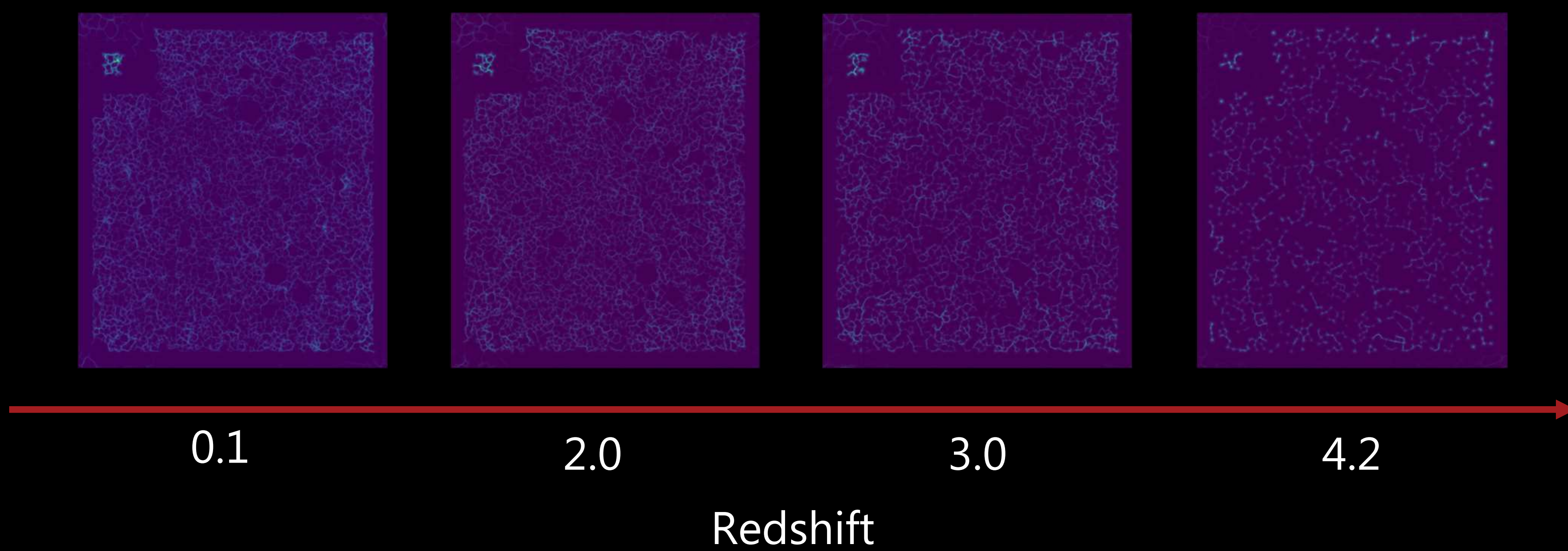


Can we determine the redshift range we are looking at?

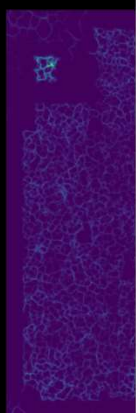
Idea: The structure of the dark matter will look different at higher redshifts

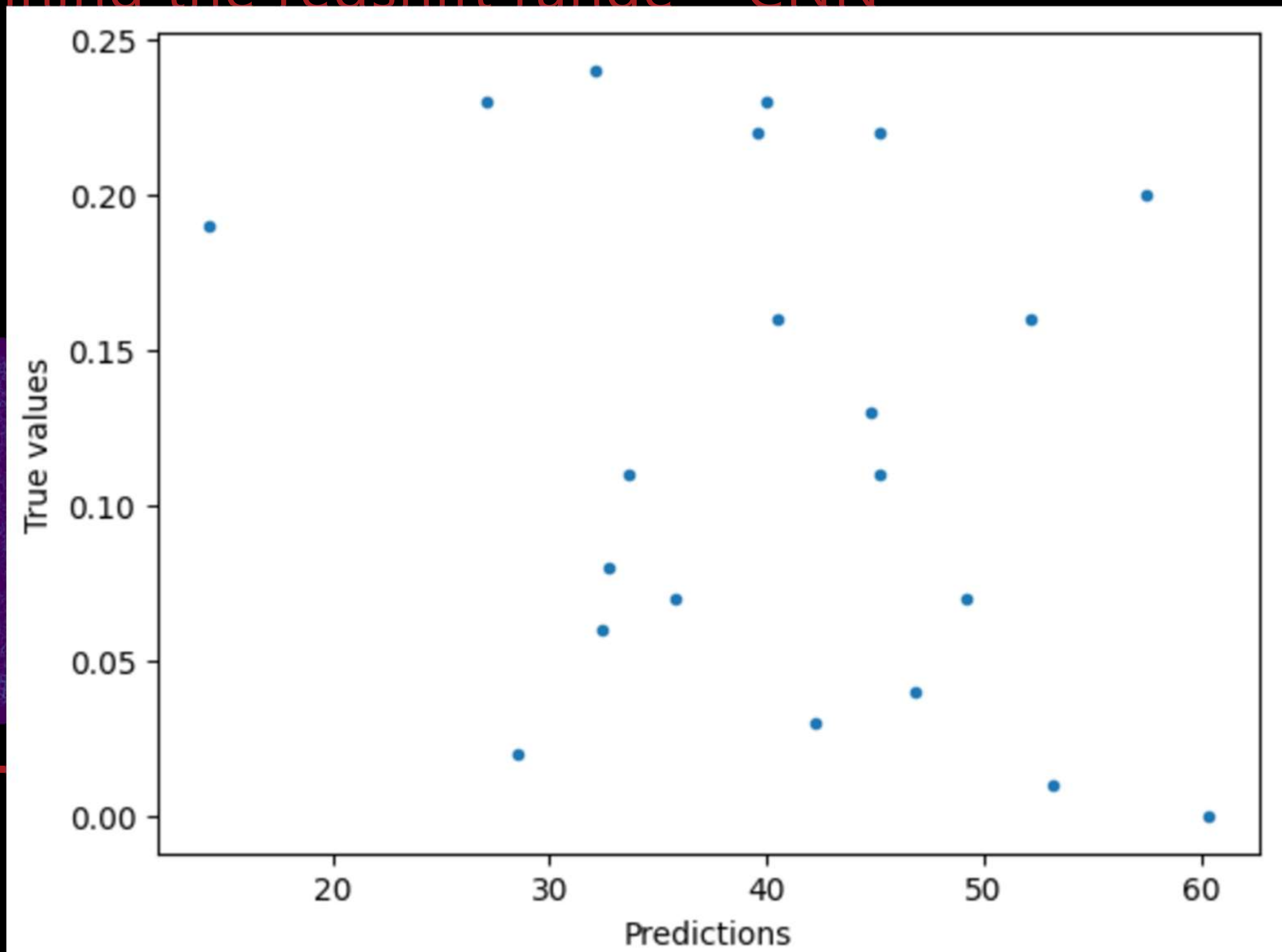


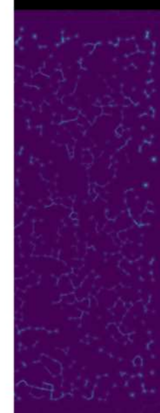
Determining the redshift range – CNN



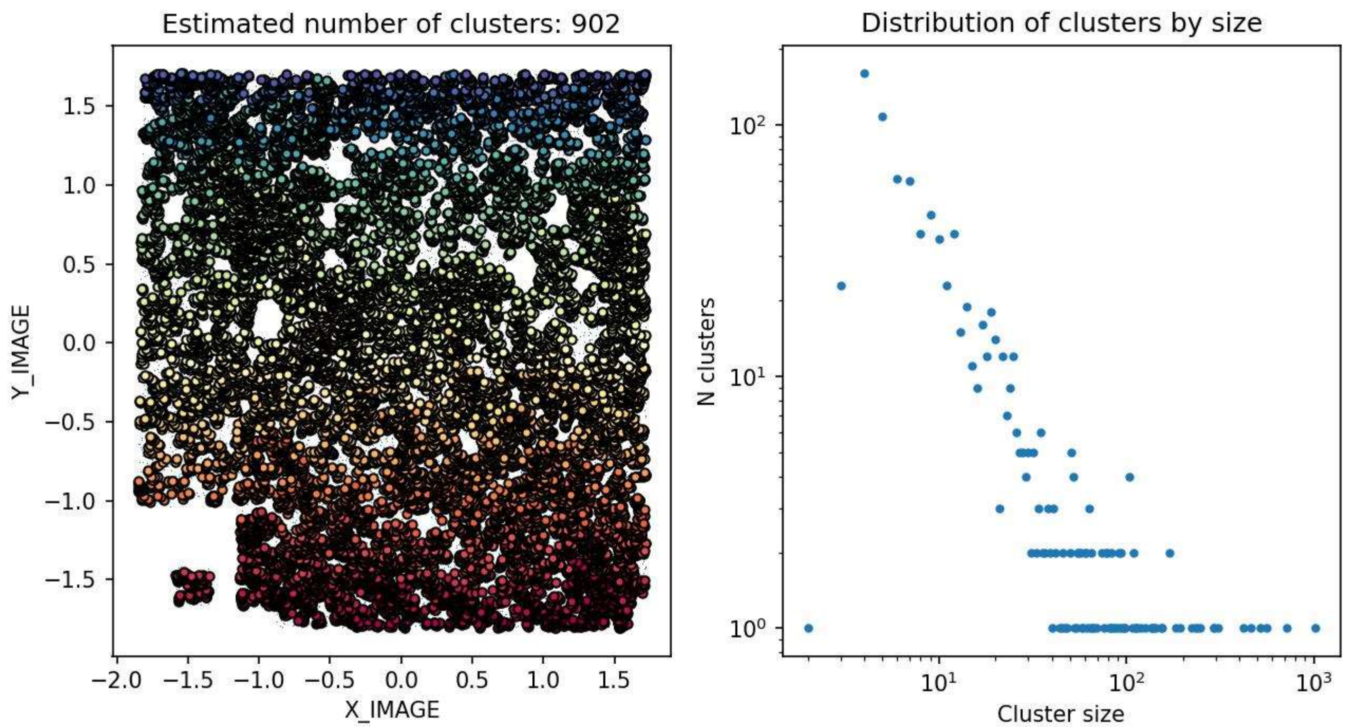
Determining the redshift range – CNN



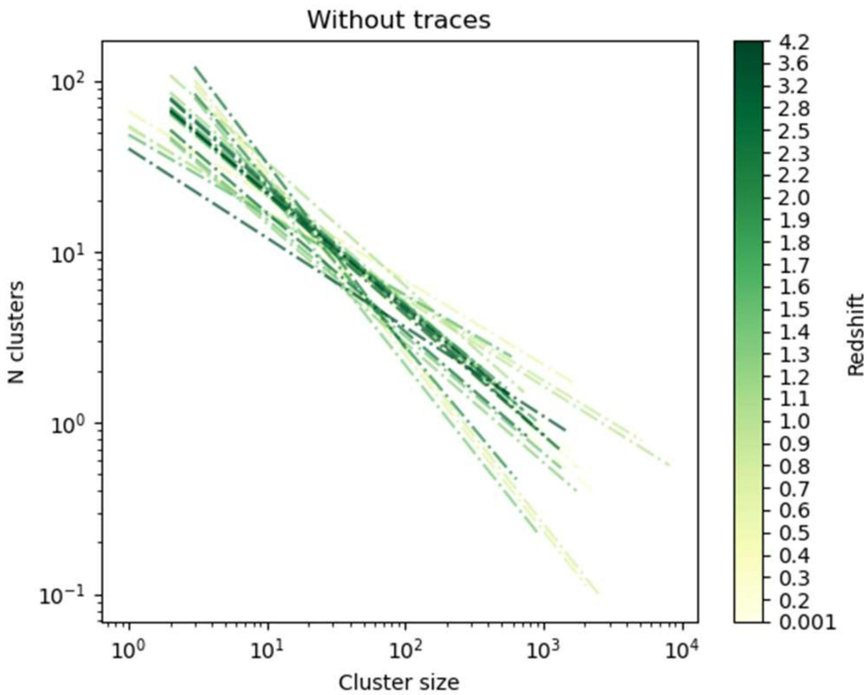


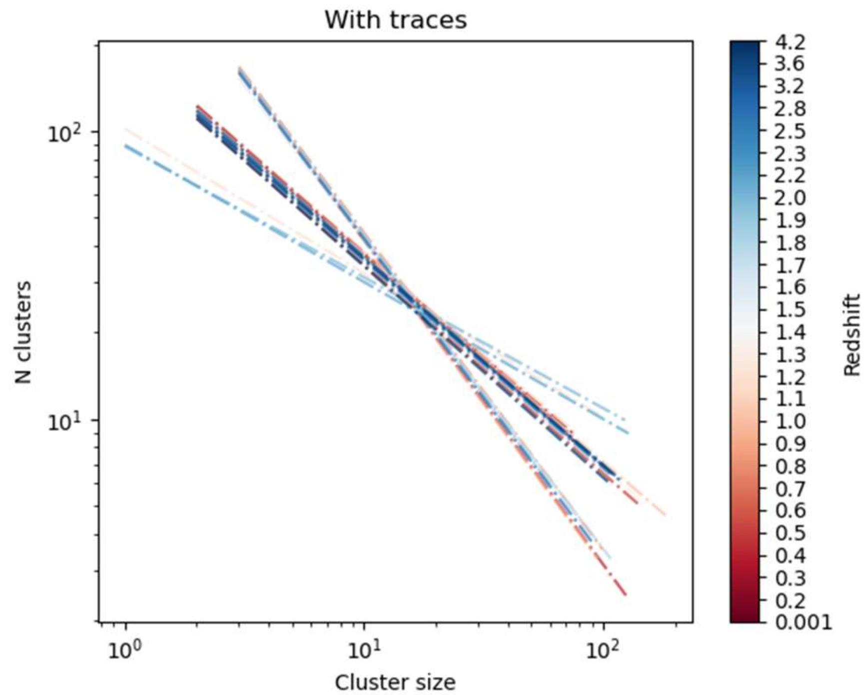


Determining the redshift range - Clustering

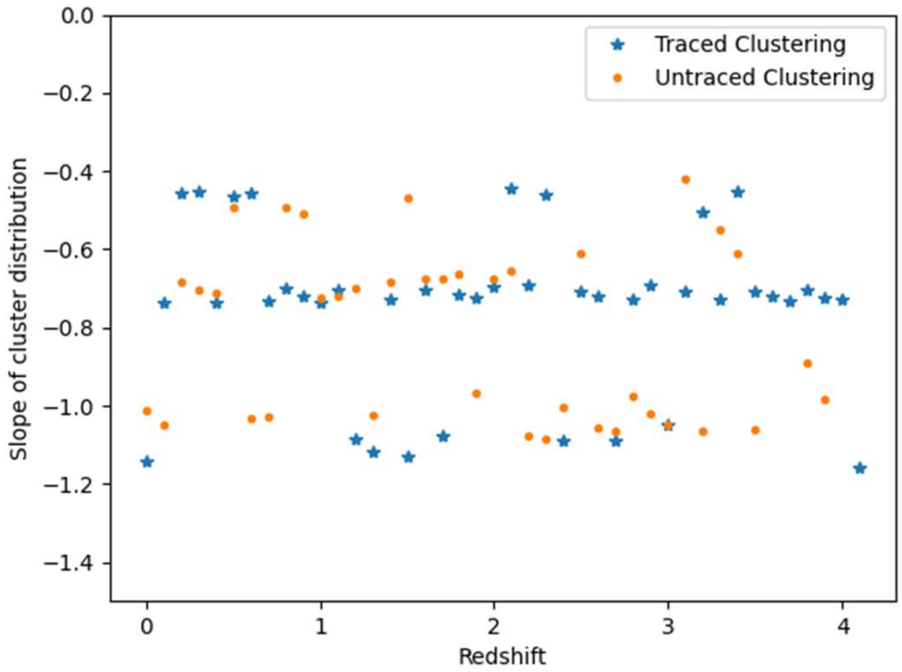


Determining the redshift range - Clustering





Determining the redshift range - Clustering



Conclusion & future research

- Generated a model of Dark Matter
- ML on the model gave no further insight
- A way to verify MCPM-generated model
- Create a 3D model
- More data for CNN

APPENDIX

APPENDIX - Overview

- Data processing
- Machine Learning Algorithms
  - Neural Network
  - Boosted Decision Tree
  - DBSCAN clustering
  - CNN

APPENDIX - Data processing

The choice of making redshift a small range was made so we kept the image as close to 2D as possible

The variable “ANT” (Average of Nearby Trace) was made after we saw that trace was not enough information for our ML algorithms.

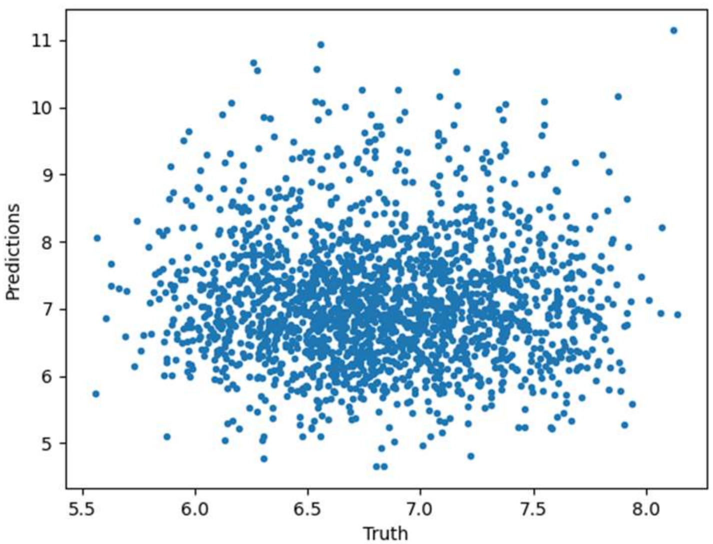
APPENDIX – Machine Learning algorithms
Tensorflow Neural Network

We split the data into two parts. One for training (75%) and the other 25% were for testing.

The NN was optimized using Bayesian Optimization, Grid search was tried but gave worse results for hyperparameters.
Data points: 16303
Redshift: 0.001 – 0.2

Best hyperparameters: {
'batch_size': 24.0,
'learning_rate': 0.00893480474068008,
'num_epochs': 38.0,
'num_layers': 1,
'num_neurons': 109.0}

We then got a MSE = 1.198



APPENDIX – Machine Learning algorithms
Tensorflow Neural Network

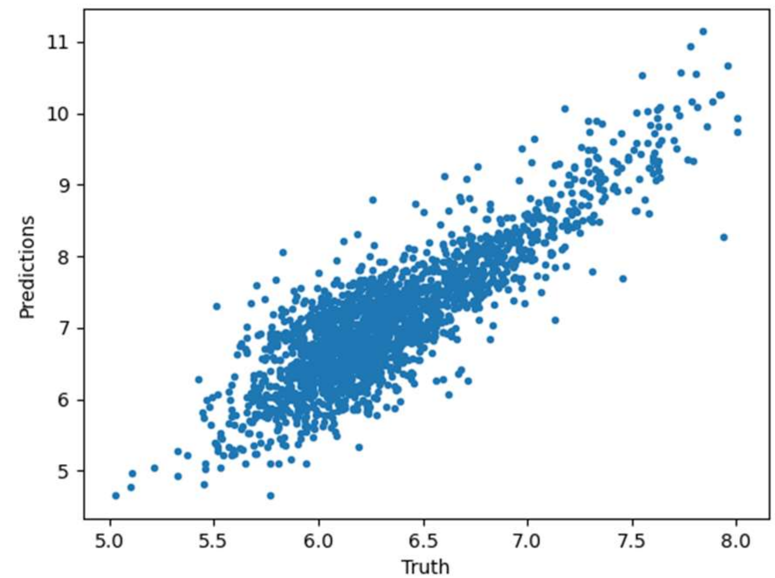
After we saw the result of the NN, we decided to add light as a variable to test if it was our model or the data that was bad. Light is a good estimator for mass, because there is a coloration between them called Light-to-Mass ratio.

The NN was optimized with Bayesian Optimization.

Data points: 16303
Redshift: 0.001 – 0.2

Best hyperparameters: {
'batch_size': 24.0,
'learning_rate': 0.00831077138618366,
'num_epochs': 49.0,
'num_layers': 1,
'num_neurons': 95.0}

We then got a MSE = 0.570

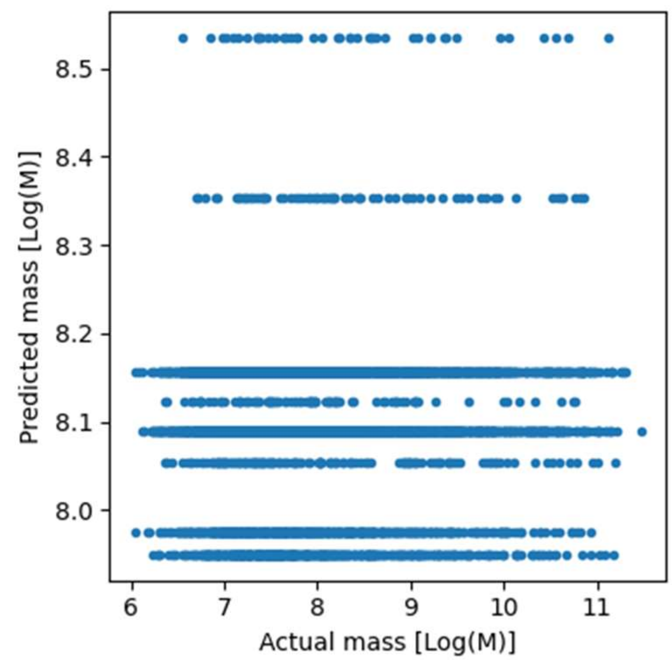


APPENDIX – Machine Learning algorithms SKLearn DecisionTreeRegressor

We split the data into two parts. One for training (75%) and the other 25% were for testing.

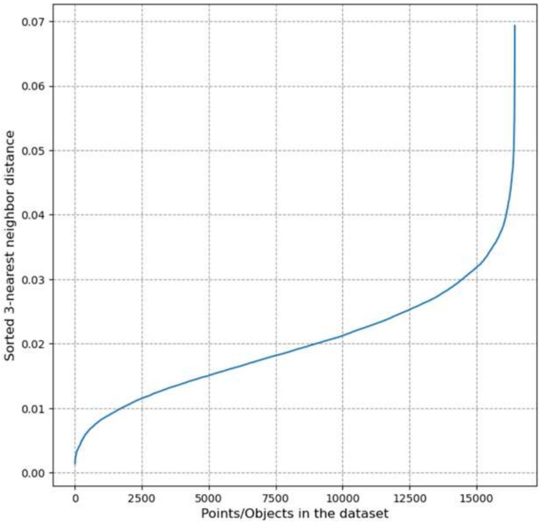
The BDT was optimized using Bayesian Optimization.
Values for HPO:
Data points: 26223
Redshift: 0.3 – 0.4
Bayesian Optimization
Best hyperparameters: ([
('max_depth', 3),
('min_samples_leaf', 100),
('min_samples_split', 94)])

We then got a MSE = 1.047



APPENDIX – Machine Learning algorithms SKLearn DBSCAN clustering

We first used scoring functions Silhouette score and DBCV index to tune hyperparameters epsilon and min samples. This resulted in only a small part of the data being clustered in very few clusters.
Our goal was to cluster most of the data to get a measure of the structure for various redshifts. We therefore created a measure of nearest neighbors and used the epsilon where most of the data can be clustered. Then we tuned min samples.



Here we chose epsilon = 0.03

APPENDIX – Machine Learning algorithms TensorFlow CNN

```
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=images[0].shape),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(96, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)  # Output layer for regression task
])

optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3)
model.compile(optimizer=optimizer, loss='mean_squared_error')
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

start_time = time.time()
model.fit(x_train, y_train, epochs=20, validation_data=(x_test, y_test), callbacks=[stop_early], verbose=1)

loss = model.evaluate(x_test, y_test)
```