DOP Project Report

Searching for Dark Matter Using Machine Learning Techniques

Vaibhav Chaudhari
2017B5A70834G

A report presented for the project work of
Semester-1 of 2020/2021

BITS Pilani
K K Birla Goa Campus

Physics Department
BITS Pilani K.K Birla Goa Campus
2/12/2020

1 Abstract

The field of Astronomy has experienced a rapid growth, both in terms of the data size and complexity and also in terms of the new discoveries and findings. One of the best examples of how the field of Astronomy has grown can be Dark Matter. Dark matter is a form of matter thought to account for approximately 85% of the matter in the universe. Machine Learning algorithms can be very useful in analyzing the data obtained from various sources and can be used for prediction of helpful results. In this report, I have used the Dataset containing the Dark Matter Halo Properties to predict empirically the properties of the galaxies. Many Supervised Learning methods like Random Forest, Support Vector Machine, Stochastic Gradient Descent have been used to get the optimum results. Results after using Dimensionality Reduction and PCA are also mentioned in this report.

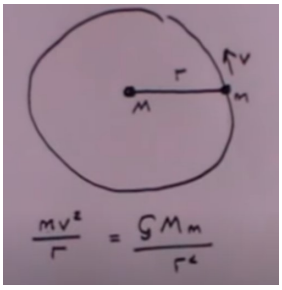
.

2 Introduction

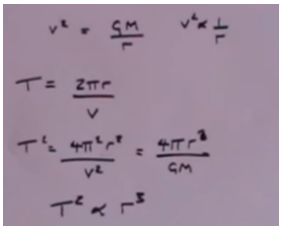
The visible universe is made of protons, electrons and neutrons bundled together into atoms. But this visible matter makes up only 5% of the mass of the universe. The rest of the mass is thought to be made of an invisible substance known as Dark Matter and Dark Energy(a force that repels gravity).

The amount of visible matter in the galaxies cannot account for their shapes, movements, distribution etc. This can be reasoned as the speed with which the galaxies are rotating as opposed to the gravity generated by the observable matter cannot hold the galaxies together. The galaxies should have been torn apart a long time ago. This proves the fact that there is something present in these galaxies which gives them extra mass which in turn generates the extra gravity which is needed by them to stay intact. This unknown and invisible matter is called "Dark Matter".

Let us understand this practically. Assume a planet of mass **m** moving around the star of mass **M** with velocity **v** at a distance **r** from the center. The Centripetal force in this case is provided by the gravitational force and so



Since **G** and **M** are constant, v^2 is proportional to 1/r, the farther away a planet is from the star, the smaller is its orbital velocity. This leads to Kepler's third law.



Now if we were to plot the graph between **v** and r of our solar system, we will find that the further out we go the lesser is the orbital velocity. If we assume that a planet is revolving inside the sun then the **M** value in the formula changes and the velocity reduces until it reaches zero at the center. Now let us move onto the Galaxy rotation. If a star is moving with a velocity v then we would expect the same kind of graph that we got for our solar system but the type of graph that we observe is that after a certain point, the velocity becomes almost constant. If you go to the formula of v^2=GM/r, since **v** becomes constant and G is also a constant, the value of M/r should also remain constant. Now if we think normally, if we move towards the outside of the Galaxy obviously the mass M will increase as a greater number of stars are accumulated the farther we move from the center of the Galaxy. If we move to the edge of the Galaxy, the mass contained within the orbit barely changes at all so we would expect that even if it was true for the inner stars by the time we reach to the outer stars, the mass would barely change at all and their velocity will start to decrease but we observe that it remains constant. And even if the velocity of the other stars is what it is measured, the stars would not have enough centripetal force to keep them in an orbit and the Galaxy should just fly off because there isn't enough gravitational force. Therefore, we come to the conclusion that the mass of the Galaxy must be greater than the observed mass of the Galaxy which is due to presence of Dark Matter.

3 Aim of the Project

The aim of this project is to understand how the Dark Matter plays an important role in the functioning of the universe and to also use various Machine Learning techniques in order to extract relevant information about the galaxies. The extraction of these properties like the Stellar Mass of a galaxy helps us to determine the accuracy of the data we have collected as there might be some discrepancies.

4 Dark Matter

There are two suggested explanations for the discrepancy between the luminous mass and the conventional dynamical mass of galaxies: dark halos and the modified Newtonian dynamics (MOND).

4.1 Dark Halo Model

Dark Matter Halo is a hypothetical region which contains gravitationally bound matter and has decoupled from cosmic expansion. A dark matter halo may contain multiple clumps of Dark Matter bound together by gravity called subhalos. The dark matter halo of a galaxy envelops the galactic disc and extends well beyond the edge of the visible galaxy.

The presence of dark matter in the halo is understood from its gravitational effect on a spiral galaxy's rotation curve. The rotational velocity of the galaxy would decrease at large distances from the galactic center without large amounts of mass throughout the halo. Observations of spiral galaxies show that the rotation curve of most spiral galaxies flattens out, meaning that rotational velocities do not decrease with distance from the galactic center. The absence of any visible matter to account for these observations implies that dark matter exists. A model of the spiral galaxy halos is developed which can match the observed 'flat' rotation curves of some galaxies.

In the dark halo model, we assume that the halo is spherical and has a density distribution which is approximately equal to that of an isothermal sphere. There are three parameter fits: M/L for the visible matter, the central halo density ρ_0 , and the halo core radius r_c.

$$\rho(r) = \rho_0 \left[1 + \left(\frac{r}{r_c} \right)^2 \right]^{-1}.$$

4.2 MOND Model

Modified Newtonian dynamics (MOND) is a hypothesis which proposes a modification in Newton's laws to make sense of the observed properties of galaxies. The discrepancy of larger speeds of the stars in the galaxy could be resolved if gravitational force came to vary inversely with radius as opposed to the inverse square of the radius, as in Newton's law of gravity or alternatively if the gravitational force experienced by a star in the outer regions of a galaxy was proportional to the square of its centripetal acceleration as opposed to the centripetal acceleration itself, as in Newton's second law. Violation of Newton's laws in the MOND model occurs at extremely small accelerations.

5 Machine Learning

Machine Learning Algorithms are very useful in Astronomy and with the growth in the size of data and complexity of it, it has become an irreplaceable tool. I have used Machine Learning Techniques in this report in order to obtain the best possible values for the galaxy properties which are crucial in finding out things about the galaxy. I have taken the dataset for this project from here

5.1 Dataset

The dataset that was obtained from the above link was in the HDF5 file format. An HDF5 file is a container for two kinds of objects: datasets, which are array-like collections of data, and groups, which are folder-like containers that hold datasets and other groups. In order to use this dataset, I had to first import the dataset in the H5PY format ans then afterwards convert that into a dataframe using Pandas Library in Python.

Impoting the Dataset in the H5PY fromat

```
In [79]: f = h5py.File('galaxies.Z00.h5','r')

In [80]: f.keys()
Out[80]: <KeysViewHDF5 ['Galaxies']>

In [94]: list( f.keys() )
Out[94]: ['Galaxies']
```

Using the h5py file using Pandas

```
In [105]: df = pd.DataFrame(f['Galaxies'][:])
          df=df.iloc[:,0:9]
          df=df.iloc[:20000, :]

In [106]: df.head()
```

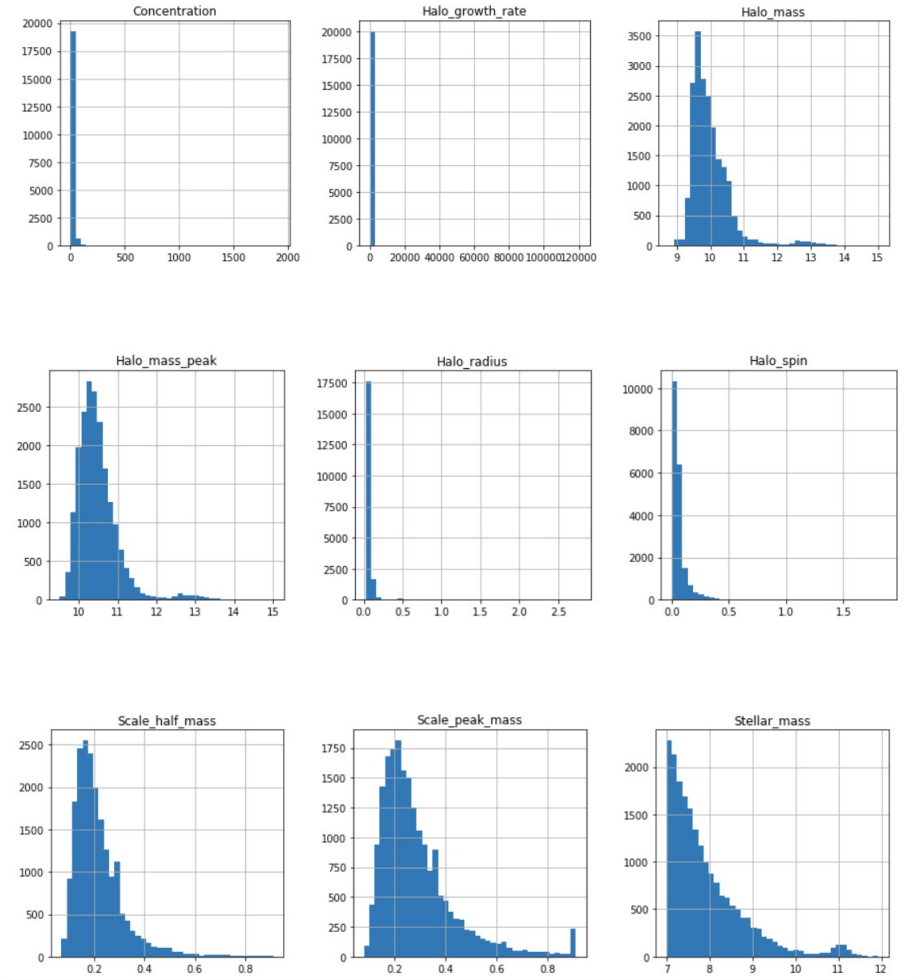
Out[106]:

	Halo_mass	Halo_growth_rate	Halo_mass_peak	Scale_peak_mass	Scale_half_mass	Halo_radius	Concentration	Halo_spin	Stellar_mass
0	11.061280	0.0	11.609557	0.70	0.52	0.133524	17.259012	0.04139	9.965789
1	9.765779	0.0	10.594986	0.46	0.16	0.049396	26.610495	0.05848	7.229853
2	9.743457	0.0	10.154882	0.24	0.19	0.048560	38.355480	0.05791	7.299570
3	9.640841	0.0	10.409262	0.35	0.30	0.044880	15.613451	0.02367	7.332184
4	10.748060	0.0	11.260630	0.69	0.64	0.105389	19.983772	0.04501	9.357066

5.2 Features and Correlation

5.2.1 Inspecting the Features in the Dataset

The first thing I did after getting the dataframe of the dataset was to check for null values and remove them in order to remove the discrepancy from the data. After this, to visualize the features of the dataset and to extract some information out of it, I plotted Histograms for the various features present in the dataset. The histograms are shown below:



The following conclusions can be drawn from the above histograms:

- Halo_growth_rate, Halo_radius, halo_spin, Concentration, scale_peak_mass, and stellar_mass do not follow Gaussian distribution. Because of this we will be using a min-max scalar for them.
- halo_mass_peak, Halo_mass and scale_half_mass approximately follow the Gaussian Distribution, therefore, I will use a standard scalar for feature scaling.
- Halo_growth_rate, halo_radius, Concentration and halo_spin remains approximately constants as stellar mass increase. Hence we expect a small correlation between these features and stellar mass.

5.3 Pre-processing

In order to make the execution of the code faster and to increase the resuability of the code, we have used pipelines. A machine learning pipeline is used to help automate machine learning workflows. They operate by enabling a sequence of data to be transformed and correlated together in a model that can be tested and evaluated to achieve an outcome. I have made two separate pipelines. One for the features using the Min-max scaling and another for features using the Standard Scaling. I have then combined these pipelines into a single Pipeline for the ease of using it.

In [134]:
```
class featureSelection(BaseEstimator, TransformerMixin):

    def __init__(self, featureList):
        self.featureList = featureList

    def transform(self, X):
        return X[self.featureList].values

    def fit(self, X, y=None):
        return self
```

In [140]:
```
minmaxFeature=["Concentration", "Halo_growth_rate","Halo_radius", "Halo_spin", "Scale_peak_mass" ]
standardFeature= ["Halo_mass", "Halo_mass_peak", "Scale_half_mass"]
```

In [141]:
```
#Pipeline for minmax Scaling
minmaxStep= [("minmax_selector", featureSelection(minmaxFeature)),
             ("simpleImputer", SimpleImputer(strategy = 'mean')),("minmax_scaler", MinMaxScaler())]
minmaxPipeline= Pipeline(minmaxStep)
```

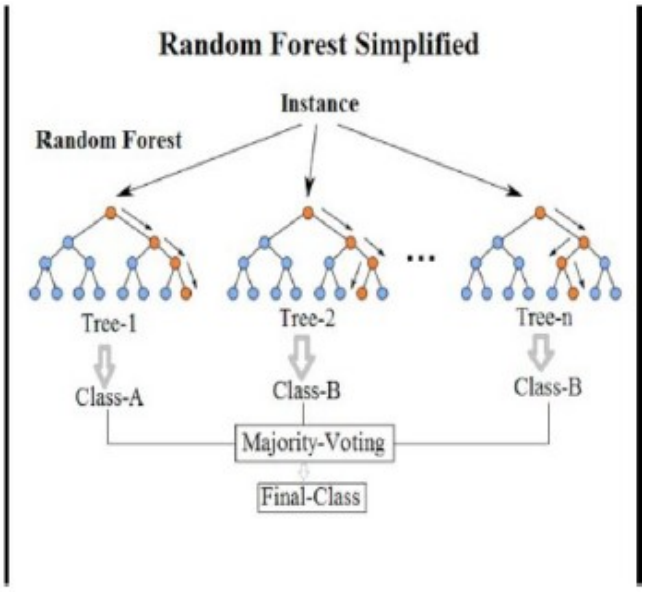
In [142]:
```
#Pipeline for Standard Scaling
standardStep= [("standard_selector", featureSelection(standardFeature)),
             ("simpleImputer", SimpleImputer(strategy = 'mean')),("standard_scaler", StandardScaler())]
standardPipeline= Pipeline(standardStep)
```

In [143]:
```
#Pipeline for combined output from the above pipelines
finalPipeline=FeatureUnion([("minmax", minmaxPipeline), ("standard", standardPipeline)])
```

5.4 Machine Learning and Deep Learning Algorithms

5.4.1 Random Forest Classifier

A brief explanation of the Random Forest algorithm comes from the name.Rather than utilize the predictions of one decision tree, the algorithm will take the ensemble results of an outsized number of decision trees (a forest of them).The "Random" a part of the name comes from the term "bootstrap aggregating",or "bagging". What this suggests is that every tree within the forest only gets to train on some subset of the complete training dataset (the subset is decided by sampling with replacement). The elements of the training data for every tree that are left unseen are held "out-of-bag" for estimation of accuracy. Randomness also helps decide which feature input variables are seen at each node in each decision tree. Once all individual trees are fit a random subset of the training data, employing a random set of feature variable at each node, the ensemble of all of them is employed to give the final prediction.

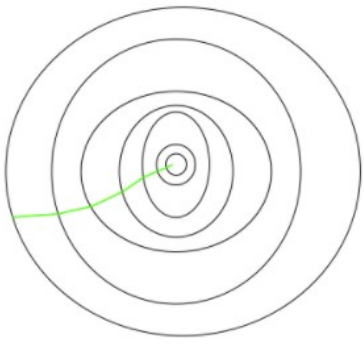


5.4.2 Stochastic Gradient Descent

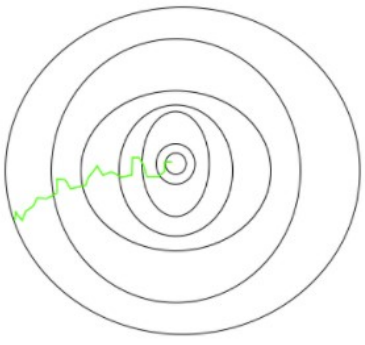
Gradient Descent is an important optimization technique in ML and DL. It can be used with most of the learning algorithms. Gradient means the slope of a function, which measures the degree of change of a variable with respect to the changes of another variable. Mathematically, Gradient Descent is a convex function. Greater the gradient, steeper the slope is. From an initial value, gradient descent is run iteratively to find the optimal values of the parameters such that they can minimize the cost. Typically, there are three types of Gradient Descent:

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent

In this project we are using Stochastic Gradient Descent or SGD. The word 'stochastic' is linked with the process associated with random probability. So in stochastic gradient descent,for each iteration a few samples are selected randomly rather than using the whole dataset. In Gradient Descent, batch" denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In Batch Gradient Descent, the batch is taken to be the whole dataset.If we use the whole dataset we will reach the minima in a less noisy/random manner. But a problem arises when our database is very big, like a million samples in your dataset.So if we use normal gradient descent we will do calculations for one million samples to complete one iteration of Gradient Descent, and this will be repeated until the minima is reached. So this process is computationally very expensive to perform. This problem is solved by SGD where it uses batch size of 1, i.e only a single sample to perform each iteration. The samples are randomly shuffled and then selected for performing each iteration. So, in SGD, we find out the gradient of the cost function with respect to a single example instead of the sum of the gradient of the cost function for all the examples. Since we use only one example at random from the dataset for each iteration, the path taken by it to reach minima is noisier than the general gradient descent algorithm. It doesn't matter much though as we reach the minima with significantly shorter training time. Even though it takes more iterations to reach the minima, it is still computationally much less expensive as computation for each iteration is much much less.. Hence, in most scenarios, SGD is preferred over Batch Gradient Descent for optimizing a learning algorithm.



Path taken by Gradient descent



Path taken by SGD

5.4.3 Support Vector Model

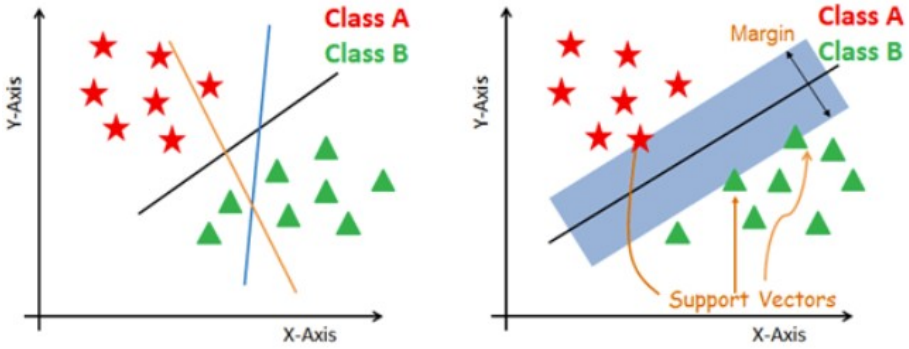
SVM classifiers separate data points using a hyperplane with a large amount of margin, so it is known as a discriminative classifier. SVM finds an optimal hyperplane which helps to classify new data points. It is considered as a classification approach, but can be used in both types of classification as well as regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate various classes. SVM generates optimal hyperplanes in an iterative manner, which minimizes error. Core idea here is to find a maximum marginal hyperplane(MMH) that divides the dataset into classes.

Support vectors are the data points, closest to the hyperplane. These points will define the separating line better after calculating the margins. These points are important in construction of the classifier. A hyperplane is a decision plane which separates between a set of objects which are in different classes.

A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest points. If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

The main objective is to segregate the given dataset within the absolute best way. The space between the either nearest points is understood because the margin. The target is to pick a hyperplane with the utmost possible margin between support vectors within the given dataset. SVM searches for the utmost marginal hyperplane within the following steps:

- Generate hyperplanes which segregate the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.
- Select the right hyperplane with the maximum segregation from the either nearest data points as shown in the right-hand side figure.

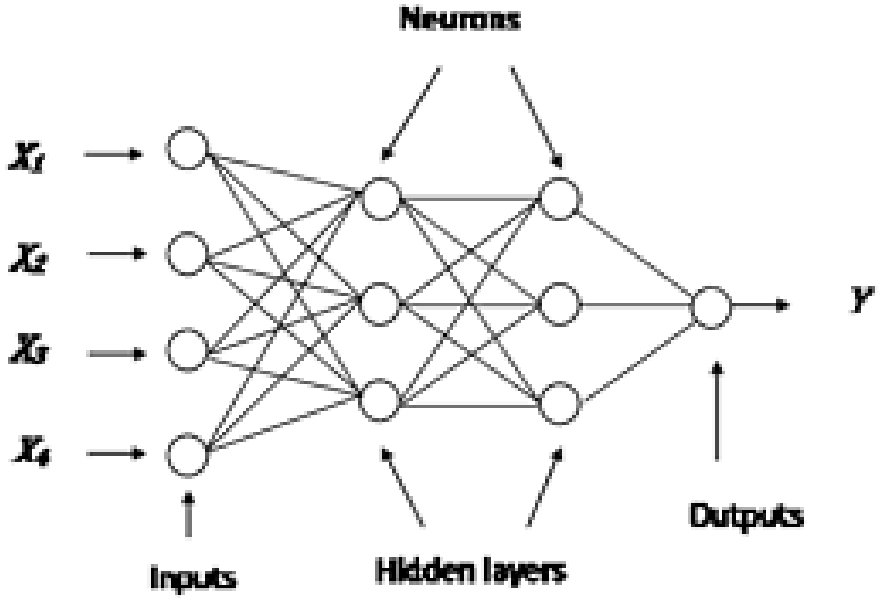


The SVM algorithm is implemented in practice employing a kernel. A kernel transforms an input file space into the specified form. SVM uses a way called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a better dimensional space. In other words, you'll say that it converts non separable problems to separable problems by adding more dimension thereto . It is most useful in non-linear separation problem. Kernel trick helps you to create a more accurate classifier.

5.4.4 Multilayer Perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network (ANN).

An MLP consists of a minimum of three layers of nodes: an input layer, a hidden layer and an output layer. Aside from the input nodes, each node could be a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that's not linearly separable.



5.5 Regression using Random Forest, SVM, SGD

5.5.1 Model Selection using cross-validation

We are applying cross validation on each of our above mentioned model. The cross validation score comes in terms of the negative mean square error. We are using the Regressor for each of the above mentioned models to find the negative mean squared error. The cross validations splits the dataset into a number of subsets(here split into 5 subsets). This has been done to prevent overfitting in the model which is one of the major issues while using Machine Learning Models.

In [154]:
```
#Random Forest Regression
RandomForestSteps=[("prep", finalPipeline), ("RandomForest", RandomForestRegressor())]
RandomForestRegressor=Pipeline(RandomForestSteps)

CrossValidation_score=cross_val_score(RandomForestRegressor, X_train, y_train, cv=5, scoring="neg_mean_squared_error")
displayScores(CrossValidation_score)
```

```
Scores: [-0.05021755 -0.04521919 -0.04716018 -0.04723301 -0.04888512]
Mean Score: -0.047743010663433436
Standard Deviation of Scores: 0.0016968938729558436
```

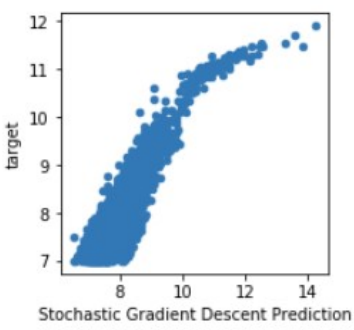
In [155]:
```
#Support Vector Regression
SupportVectorSteps=[("prep", finalPipeline), ("SupportVector", SVR())]
SupportVectorRegressor=Pipeline(SupportVectorSteps)

CrossValidation_score=cross_val_score(SupportVectorRegressor, X_train, y_train, cv=5, scoring="neg_mean_squared_error")
displayScores(CrossValidation_score)
```

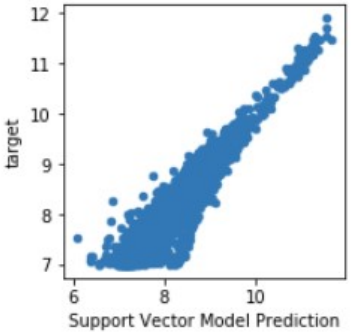
```
Scores: [-0.05741126 -0.05861074 -0.05620758 -0.0578667  -0.05962345]
Mean Score: -0.057943946203690276
Standard Deviation of Scores: 0.0011463002260517167
```

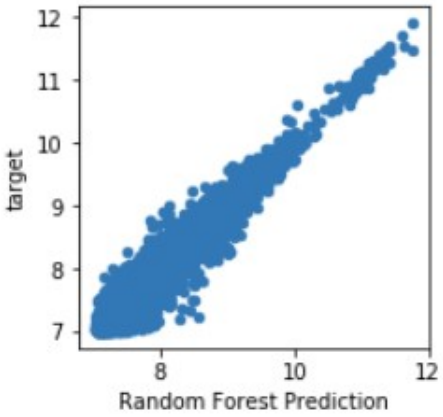
In [156]:
```
#Stochastic Gradient Descent Regression
StochasticGradientDescentSteps=[("prep", finalPipeline), ("StochasticGradient", SGDRegressor())]
StochasticGradientDescentRegressor=Pipeline(StochasticGradientDescentSteps)

CrossValidation_score=cross_val_score(StochasticGradientDescentRegressor, X_train, y_train, cv=5, scoring="neg_mean_squared_erro
displayScores(CrossValidation_score)
```

```
Scores: [-0.13140428 -0.13478944 -0.13225921 -0.12773834 -0.13534038]
Mean Score: -0.13230632890516827
Standard Deviation of Scores: 0.0027222679813450616
```

We then train all the three models on the dataset with a split of 80% training data and 20% testing data and plot the charts to get a better understanding of which model is better suited for our work based on visual inspection.







As we can clearly see from the plots above that the Random Forest gives us the best CV score among the three models and therefore we will focus majorly on Random Forest only after this.

5.5.2 Hyperparameter Search

There are different parameters for every model, and the GridSearch library runs the model for each and every parameter and tells us which parameter is optimum for the use. Like in the code shown below, we get the value for the max_depth parameter in the random forest model as 12.

Hyperparameter Search using Cross-Validation

```
In [166]: from sklearn.model_selection import GridSearchCV
          ff = [{"RandomForest__max_depth": range(1,30)}]
          GSCrossValidation=GridSearchCV(RandomForestRegressor, ff, scoring = 'neg_mean_absolute_error')
          GSCrossValidation.fit(X_train, y_train)
          print(f"Best Max_dept : {GSCrossValidation.best_params_}")

          Best Max_dept : {'RandomForest__max_depth': 12}
```

The final evaluation of the model is done using the parameter of Random Forest obtained using GridSearch. The predicted RF values are then compared to the target values and we observe that both the values are approximately the same.

	target	RF Prediction
19134	7.238688	7.318531
4981	7.735857	7.764981
16643	7.497379	7.480930
19117	7.328487	7.122101
5306	7.121845	7.124358

5.6 Regression using Neural Networks

The performance of MLP is compared with the 3 ML models and we observe that the results obtained from Neural Network is not better.

```
from sklearn.neural_network import MLPRegressor

MLP_steps=[("prep", prep_pipe), ("MLP", MLPRegressor(hidden_layer_sizes=
(6,5,)))]
MLP_regressor=Pipeline(MLP_steps)
CV_MLP_score=cross_val_score(MLP_regressor, X_train, y_train, cv=5, scor
ing="neg_mean_squared_error")
display_scores(CV_MLP_score)
```

```
Scores: [-0.0702481  -0.07027606 -0.0790737  -0.0900264  -0.07810145]
Mean: -0.07754514033034683
Standard Deviation: 0.007273431427737025
```

5.6.1 GridSearch for Best Number of Hidden Units

We use the GridSearch library for finding the best number of hidden layers in the neural network so that our neural network can work efficiently and we find that:

```
pg_MLP= [{"MLP__hidden_layer_sizes": [(6, 5,), (6, 4,), (6,3,),(6,2,),
(5,4,),(5,3,), (5,2,), (4,6,)]  } ]

GSCV_MLP=GridSearchCV(MLP_regressor, pg_MLP, scoring = 'neg_mean_absolut
e_error')
GSCV_MLP.fit(X_train, y_train)
print(f"Best Max_dept : {GSCV_MLP.best_params_}")
```

```
Best Max_dept : {'MLP__hidden_layer_sizes': (6, 5)}
```

5.7 Feature Selection and Dimensionality Reduction

5.7.1 Feature Selection

The features have been selected on the basis of the importance of each and every feature. Using the following results we can tell the feature importance of every feature in the dataset.

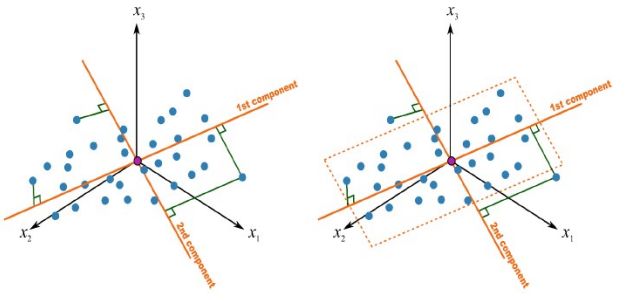
	Feature	Imp Score
0	Halo_mass	0.008864
1	Halo_growth_rate	0.000174
2	Halo_mass_peak	0.011378
3	Scale_peak_mass	0.007409
4	Scale_half_mass	0.124815
5	Halo_radius	0.007461
6	Concentration	0.818845
7	Halo_spin	0.021053

From the above table we can conclude that the concentration, scale_half_mass and the Halo_spin are the most important features. The importance score of the features is dependent on the Model that is being used. So it may be the case as when a different model is used apart from the Random Forest Model, the importance score of Concentration may not stay the highest as it is in this case. These scores are model dependent.

We then use these important features to get the cross validation scores again for the Random Forest Model.

5.7.2 PCA

Principal Component Analysis (PCA) is a feature extraction technique which combines our input variables in a specific way so we can drop the least important variables while we still have the valuable parts of all the variables. Each of the new variables after PCA are still independent of one another which is a very important factor because we now have variables which are not related to one another and these new and reduced number of variables make our computation more effective while keeping the accuracy intact to a particular extent.



6 Conclusion

I was able to predict the properties of the galaxy to a very accurate measure using the dataset of the dark halo. The machine learning model can be used to see how the Dark Halo model works and vice-versa. This acts as a standard to see how the dark halo model works and in correspondence with the Machine Learning Algorithm that we have used here.

Since this model works, we can apply this model to a bigger scale for eg. if the dark halo model only works for spiral galaxies but if we make a few changes to this machine learning model and make it general enough, it will be able to work for all kinds or galaxies like spiral, elliptical etc. without specifying additional requirements.

Since Machine Learning picks up on the various properties of data present in our dataset, this technique can be further worked on and perfected to get a model which will be able to predict the properties of galaxies irrespective of the type of the galaxy it is applied on.

7 Appendix

This contains all the papers that have been referred to while making this report.

- https://arxiv.org/abs/1904.07248
- https://discovery.ucl.ac.uk/id/eprint/1508539/1/Peiris_Lochner_2016_ApJS_225_31.pdf
- https://arxiv.org/pdf/1904.07248.pdf
- https://academic.oup.com/mnras/article/249/3/523/1005565

8 Code and Other Resources

The code and other resources like the LaTeX document can be found on the google drive link

9 Acknowledgement

I am very thankful to Dr. Prasanta Kumar Das (Associate Professor, Department of Physics, BITS Goa) for guiding me through this project which has led me to get a wider understanding of how Machine Learning can be used in Astronomy.
I would also like to thank him for helping me, giving advice and constant motivation for this project without which this project could not have been completed.

References

[1] Dark Matter and Galaxy Rotation [Youtube]

[2] Dark Halo Model [Online]

[3] Dark Halo [Online]

[4] MOND Model [Online]

[5] MOND: Observational Phenomenology and Relativistic Extensions [Online]

[6] HDF5 for Python [Online]

[7] Hierarchical Data Formats [Online]

[8] Importing HDF5 file [Online]

[9] Standard Scaler [Online]

[10] Min-Max Scaler [Online]

[11] Machine Learning Pipelines [Online]

[12] Pipeline in Machine Learning [Online]

[13] Understanding Random Forest [Online]

[14] Decision Tree vs Random Forest [Online]

[15] Stochastic Gradient Descent [Online]

[16] Stochastic Gradient Descent — Clearly Explained [Online]

[17] Support Vector Machines [Online]

[18] An Introduction to Support Vector Machines (SVM) [Online]

[19] Support Vector Machines Introduction [Online]

[20] Multilayer Perceptron [Online]

[21] Crash Course on MLP Neural Networks [Online]

[22] Perceptrons and MLP [Online]

[23] What is MLP? [Online]

[24] What is GridSearch? [Online]

[25] Cross Validation [Online]

[26] Feature Selection Techniques in Machine Learning [Online]

[27] What is PCA and how it is used [Online]

[28] A One-Stop Shop for Principal Component Analysis [Online]