FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

INTERNSHIP REPORT

Using Machine Learning Methods To Predict Binding Energies On The AME2020 Data.

Full Name	Student ID
João Cartaxo	2020226704

Advisor: Tuhin Malik

Coimbra, Portugal - August 01, 2022

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Contents

1 | Introduction

1.1 | Machine Learning (ML) algorithms

It is needless to say that creating a theoretical model that can predict some property of a system from a given set of predefined quantities can prove to be quite the challenge, so ML algorithms are slowly creeping in, as they help find complex patterns in what sometimes seems to be random meaningless data.

But before one can create good Models, one has to first learn how to create simple Models, to do this one should try to, just as a machine, learn by a trial and error process.

Since this was the goal of this first part of my Internship, I decided to train and test myself on a already published article ”The neutron star outer crust equation of state: a machine learning approach” [1], by attempting to replicate the values there achieved.

In this report we will be discussing 4 of the most common ML algorithms:

- Linear Regressions (LR)
- Neural Networks (NN)
- Decision Trees - Random Forests (DT - RF)
- Support Vector Machines (SVM)

All of which were trained on the AME2016 [2] data and later tested on the nucleus added to the AME2020 [3] data.

Perhaps the most important thing to do now would be clarify what specific data is being used, this becomes important as the previously mentioned data contain a lot more information than what will be required in the training and testing process.

As the input data of the ML algorithms, it was used the number of protons as well as the number of neutrons of a given nucleus, and for the output data, the targets were the binding energies.

Note that the binding energies used aren’t the ones seen in the AME2016 and AME2020 files, as they aren’t in the same units.

2 | Results

2.1 | Linear Regression

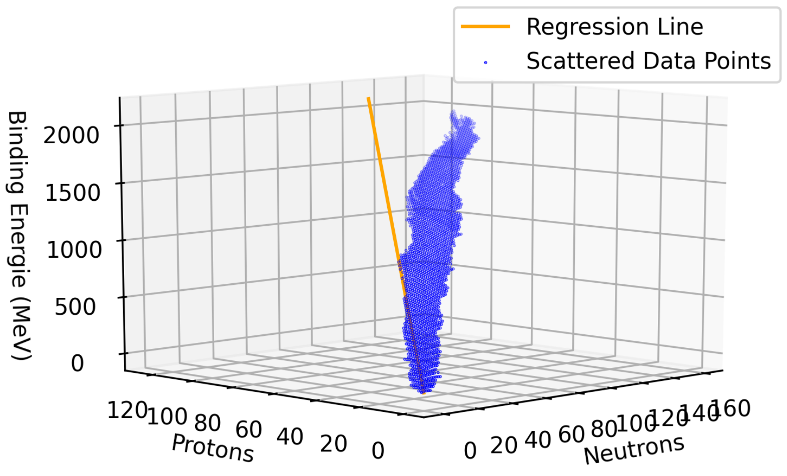
Linear Regression is by far the simplest of the ML algorithms that will be explored, and as such, it also should wield the worse of the results.

By analysing the data, we can see it doesn't want to behave linearly, and as such we should expect a fairly big loss, since the Linear Regression fails capture any of it's non-linearity.

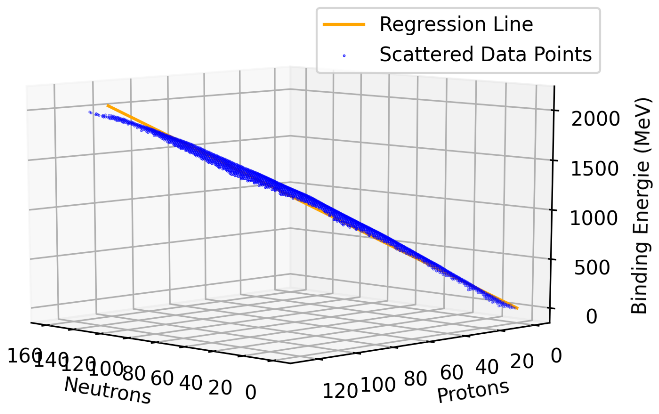
The Root Mean Squared Error (RMSE) found in the AME2016 data was ∼45 MeV, while the RMSE for the AME2020 data was ∼67 MeV.

The Following LR Model was found:

$$BE = 10.7P + 6.0N$$



(a) -135 degree 3D plot



(b) 135 degree 3D plot

Figure 1: Linear Regression Plots

2.2 | Neural Networks

Neural Networks are at the heart of Machine Learning, not only are they one of the most known ML algorithm, I actually will **Boldly** claim they are indeed the face of all Machine Learning, it would be a crime to skip over these Models.

The first thing to do is create the Neural Networks structure, since we hope to replicate the results from the article, the Neural Networks were built using the same simple structure:

$$2 \to 30 \to 1$$

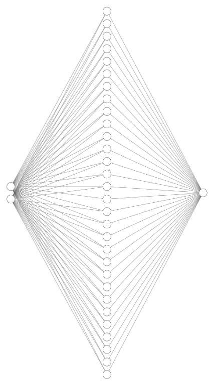


Figure 2: Simple Neural Network Structure

The training process was then divided into 5 phases:

- 1. Normalizing the data
- 2. Splitting the data
- 3. Setting the Learning Rate
- 4. Training
- 5. Predicting

The whole training process could be considered an art, by failing we can learn what works best in every data set, and so I failed, and a lot I indeed failed, but I would eventually figure out some good ways to optimize the whole process.

The first step was to Normalize[1] the input data, this meant rescaling the proton and neutron numbers, this is quite important or we will artificially be giving different weights to our inputs, and that would require some insight on the data, which goes against the process of building the Model from no previous knowledge whatsoever.

However I must note, that both the proton and neutron numbers were rescaled to be within the ranges of 0 to 100, as instead of the usual 0 to 1, this proved to be quite useful in speeding up point 4. Train .

Then we have to Split the data, this is of course important if we want to make sure our data isn't over-fitting, and specially important if we do want to test it on data it has never seen before.
Since we want both, we need to split the data into 3 sets of data, the training data, the validation data, and the testing data.
The split was done on the AME2016 data, with 70 % as training data and 30% as validation data, the testing data was the new necleus added in the AME2020 data.

[1]This was done in all of the ML algorithms, except on the simple Linear Regression.

The CSV files containing the positive experimental AME2016 and AME2020 data used can be found in the GitHub repository.

Finally we come to the last steps, we get to pick a Learning Rate and watch it go, after many attempts I came to the conclusion the best Learning Rate I could use was 0.002, if we try a bigger Learning Rate, we oscillate between losses, and never reach a good Model, if we try a smaller Learning Rate, we get ourselves into bad and deep local minima that we cannot leave from, so the best thing we can do is set a Learning Rate right in the middle, and hope to find the global minima, which ever so rarely happens! So we frequently have settle for good local minima.

As for the results, the best Model got a RMSE of 2.6 MeV on the Validation Set and a RMSE of 3.9 MeV on the Testing Set, which was quite good for such a simple structure.

However Neural Networks can go much further and get us even better results, to attempt this we can try to first use a Linear Regression, and then fit the Neural Network to the error of the Linear Model, this looks like a great idea, since it implies the Neural Networks' job is now to figure out only the non-linearity of the data, speeding the training process and starting with on a lower loss value, this could also help making a smoother plane with lesser local minima.

But one may forget what a Linear Regression looks like in a Neural Network Structure, a Linear Regression is just a single output layer

$$2 \rightarrow 1$$

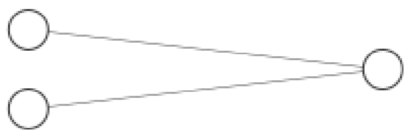


Figure 3: Linear Regression in a Neural Network Structure

Therefore we can create a new Neural Network Structure representation of the Neural Network fitted to the Linear Regression Model's error.

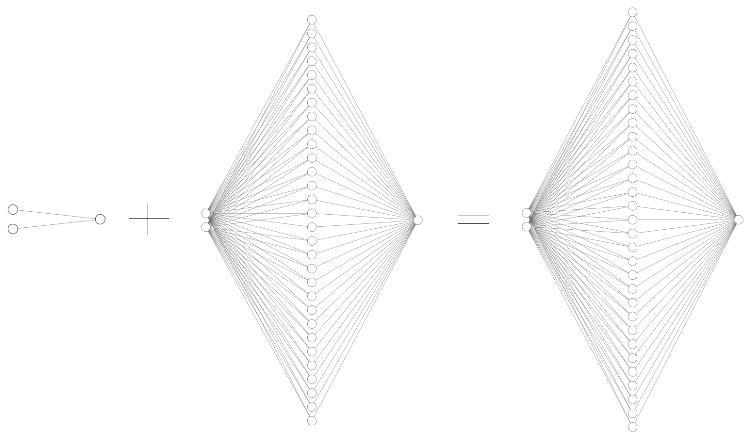


Figure 4: Neural Network With LR Structure

Although the structure seems to be the same, if we count the nodes in the hidden layer, we will find out there's a total of 31 instead of the original 30 nodes, this extra node is the entire Linear Regression.

Note however that even if the Linear Regression is just a single extra node in the hidden layer, it is assumed that the weights connecting it to the output layer are all also set to 1, with a biases of 0, this is of course due to the fact that the value calculated in the hidden layer must be the same as the one received by the output layer, otherwise the scheme of the Neural Network structure would fail to portrait the Linear Regression.

But this extra node in our hidden layer shouldn't make a big difference, and if we look at 5 and at 6 we can see a very interesting feature, they both present the same ending pattern, and this makes sense! The Simple Neural Network first learns to make a Linear Regression, and when it does, it reaches a rather stable point on the ∼38 MeV [2], but if we already give the Neural Network the Linear Regression, then we can just remove the part where the Model learns to make said Linear Regression, this is equivalent to forcing the weights and biases on the extra node to remain the same during all of the training process, which is what happens since we don't consider this additional node on our structure.

So we can conclude that the RMSE graph of the Model fitted on the Linear Regression, is nothing but a right shifted translation of the Simpler Neural Network one.

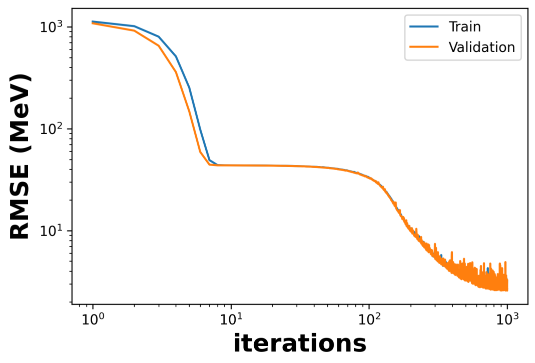


Figure 5: Simple Neural Network

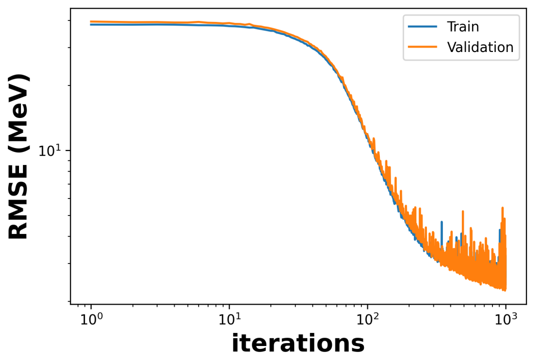


Figure 6: Neural Network on a LR Model

The Neural Network fitted to the error got a RMSE of 2.1 MeV on the validation data, and 3.5 MeV on the testing data, which although better is only due to the fact it virtually trained for more epochs on a smoother plane.

This only means we can still improve the structure of the Neural Network a lot more, possibly reaching RMSEs of less than 1 MeV on both the validation and testing data.

[2]This Value Differs from the ∼45 MeV found in the previous chapter, this is due to the fact that I allowed a Y-interception on the Linear Regression as I also allowed biases on the Neural Network

2.3 | Decision Trees - Random Forests

Decision Trees (DTs) are quite fast in their training, so you would assume they wouldn't be able to capture a lot of the non-linearity present in the data, and although this ML algorithm gives a very significant RMSE, it works much better than a Linear Regression while taking a time of equal magnitude to train.

This is fantastic if just want a very rough approximation as the training process is extremely quick.

By splitting the AME2016 data into 70% training data and 30% validation data, and using the AME2020 data, the average DT got a RMSE of 7.1 MeV for the validation data and 10.8 MeV for the testing data

The results are considerably better than the ones reached with a Linear Regression, whilst the amount of time to train remained approximately the same, nevertheless we can do much better.

By constructing a Random Forest (RF) we hope to reach a much smaller RMSE as each Decision Tree used will split the data plane differently and as the Random Forest averages them out, we should get a much better regression, which proved to be the case, by using 800 decision trees as estimators, bootstrap, and a max depth of 30, the average RF got a RMSE of 3.3 MeV for the validation data and a RMSE of 7.4 MeV for the testing data.

Yet all of these results could be much better, since the Decision Trees give better results the more information they have, by changing the training to validation ratio, we can get even better results, on both the validation and testing data.

If we instead decide to split the AME2016 data into 90% training data and 10% validating data, the Decision Tress get a RMSE 5.8 MeV of for the validation data and a RMSE of 8.9 MeV for the testing data, while the Random Forests get a RMSE of 2.3 MeV for the validation data and a RMSE of 4.6 MeV for the testing data.

But finally there's something we can still do to improve these results even further, reaching the level of Neural Networks, and this is to stack Random Forest Models, in other words, to train a Random Forest on the error of the previous Random Forest Model.

Using only 10 Random Forest iterations, including the original Random Forest trained on the actual target data, the results are significantly better than the ones achieved by a single Random Forest, reaching a RMSE of 1.4 MeV on the validation data and 3.7 MeV on the testing data while taking a fraction of the time each Neural Network takes to train.

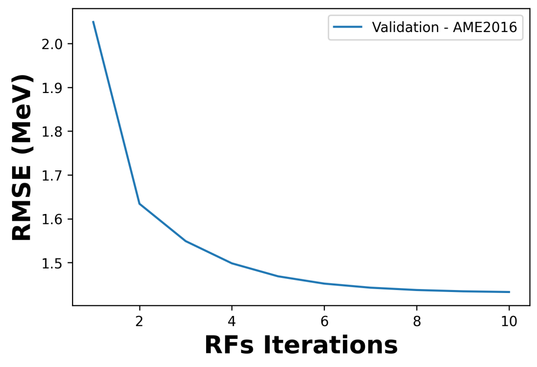


Figure 7: RF Iterations On The Validation Set

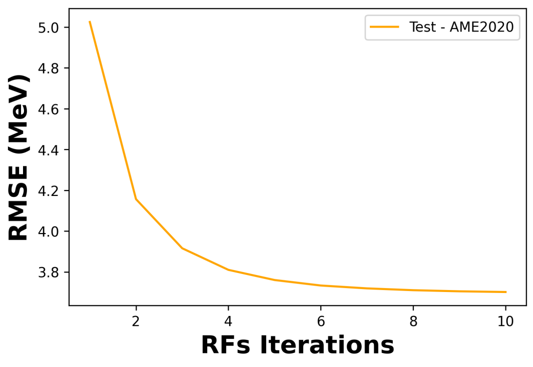


Figure 8: RF Iterations On The Testing Set

2.4 | Support Vector Machines

And so we finally reach Support Vector Machines, these will prove to be the best Models we can use for this data, since not only do they take less time than a Single Neural Network to train, but they also produce the best results by far.

Nevertheless it is important to note that this is only true if we decide to give it reasonable parameters to work with, if we just set all values randomly we could actually get ourselves into some trouble as the amount of time grows and grows, taking by far the most amount of time to train,
But again, we start by split the AME2016 data into 70% training data and 30% validation data, then we train a simple Support Vector Machine using a Gaussian Kernel with C = 80 000 , $\gamma = 0.0025$ and $\epsilon = 0.001$, reaching a RMSE of 1.35 MeV on the validation data and 1.59 MeV on the testing data, which are the lowest errors of all the base Models.

However, just like the Random Forest Models, we can stack the SVM with 10 Random Forest Models trained on the error of the previous iteration.

By stacking the SVM with the 10 Random Forests Models, each with 500 estimators and a max depth of 30, we reach the lowest RSMEs of this report, getting a RMSE of 0.66 MeV on the validation data and 0.85 MeV on the testing data, which is nothing shy of impressive.

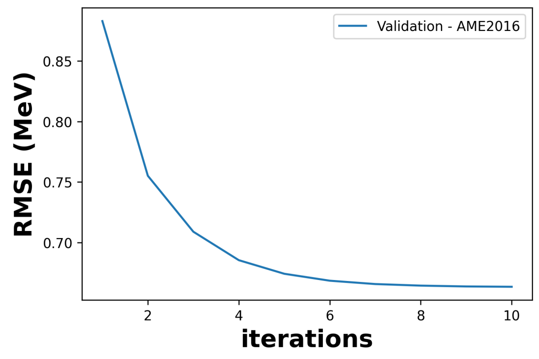


Figure 9: Support Vector Machine With RF Iterations On The Validation Set

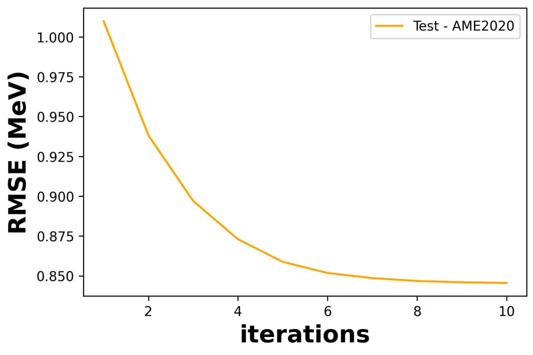


Figure 10: Support Vector Machine With RF Iterations On The Testing Set

3 | Conclusion

The Best ML algorithm for this data appears to be the SVM, not only is it fast, it also results in the lowest RMSE of all the ML algorithms used, however, one must note this is not necessarily true.

While it is true that the SVM performed better than all others, the Neural Network wasn't fully explored, a single hidden layer with 30 nodes is a very simple Model, and this could be severely improved on, by using different structures and different learning rates one should expect at least some minor improvements, not to mention, one of the Neural Networks limitations is the number of Epochs it is allowed to train, if we increase the number of Epochs and add a better structure to a new optimal Learning Rate, it is only to expect a significant drop in the RMSE.

Nevertheless the objective of the project was not to get the best possible Model, but yet to learn how to build good Models, and by comparing the results achieved in the report with the ones presented in the article, I can say it was quite the success, having replicated the results there achieved.

4 | References

[1] Utsav Murarka, Kinjal Banerjee, Tuhin Malik, and Constança Providência. The neutron star outer crust equation of state: a machine learning approach. *Journal of Cosmology and Astroparticle Physics*, 2022(01):045, 2022.

[2] Meng Wang, G Audi, FG Kondev, WJ Huang, S Naimi, and Xing Xu. The ame2016 atomic mass evaluation. *Chin. Phys. C*, 41(030003):1674–1137, 2017.

[3] Meng Wang, WJ Huang, Filip G Kondev, Georges Audi, and Sarah Naimi. The ame 2020 atomic mass evaluation (ii). tables, graphs and references. *Chinese Physics C*, 45(3):030003, 2021.