Comparing the Performance of Neural Networks and Machine Learning Models as Applied to Total Open Magnetic Flux Data*

*DATASCI 507 - Final Project

Jackson R. MacTaggart
Department of Climate and Space Sciences
University of Michigan
Ann Arbor, United States of America
jrmact@umich.edu

***Abstract*—The Sun's open magnetic flux, $B_r r^2$, and the width of the HCS-streamer belt decreased during the cycle 23-24 solar minimum compared to the cycle 22-23 minimum. In turn, total open magnetic flux (defined as the product of $B_r r^2$ and the solid angle of the open flux/non-streamer region, Ω_{CH}) was found to be conserved. In this work, we aim to implement machine learning and neural network modeling using a variety of different models to gauge the capability of predicting the total open magnetic flux (TOMF) from a plethora of related solar physics parameters. In this paper we discuss the use of decision trees (CART, random forest, and boosting models), generalized additive models (GAM), ridge regression, and a basic neural network. We find that gradient regression boosting performed the best with a MSE of 0.246. We also find that GAM partial dependencies find magnetic field magnitude, among a few other parameters, to be especially strong predictors of TOMF. Our results imply that there is a strong promise for machine learning and neural network models to be used in predicting TOMF.**

***Index Terms*—solar physics, neural networks, machine learning**

I. Introduction

1) Overview: In this project, we will use machine learning models to attempt to predict the strength of the solar cycle. First, let us establish some background. The sun goes through an 11-year, sinusoidal cycle where it goes from a period of "baseline", calm, and quiet activity (called solar minimum) to a period of higher level, volatile activity 5 ½ years later (called solar maximum). One of the open questions in solar physics is why the sun goes through this cycle and why this cycle is 11 years long (on average). Even more pressing is the question of whether the behavior of the solar cycle can be predicted.

In this project, we will use a variety of parameters to attempt to create predictions for the activity of the solar cycle. We will focus on using a quantity known as total open magnetic flux. It will be helpful for us to break down this term. In physics, flux is the amount of "stuff" traveling through some area. Total open magnetic flux (TOMF) is the amount of all the magnetic field lines flowing through some area at some distance from the sun. This has been shown to be a potentially incredibly useful quantity for predicting the behavior of the solar cycle.

2) Prior Work: Prior to 2011, the standing theory in solar physics concerning the open magnetic flux at solar minimum was that the solar minimum should represent the "baseline" level of activity for the sun and the open magnetic flux should remain constant across the minima. However, this was found to vary. In fact, from the cycle 23 minimum in 1998 to the cycle 24 minimum in 2009, the open magnetic flux was found to drop by about 40%. For a quantity that is supposed to remain constant (or relatively so) from solar minimum to solar minimum, this was a large change and a contradiction to the standing theory. In 2011, Zhao and Fisk modified how open magnetic flux is defined and found that this new definition open magnetic flux to be conserved from solar minimum to solar minimum [1]. Their new theory was validated for the cycle 22, 23, and 24 solar minima and the results were published in the Solar Physics journal. In 2024, MacTaggart, Zhao, Lepri, and Fisk (unpublished) validated this theory for the most recent solar cycle 25 minimum.

In addition to validating the most recent solar minimum, the 2024 work showed in brief data analysis that a quantity known as the "solid angle of the open flux region" could act as a predictor for the strength of the following solar cycle. This was not truly followed up on until recently.

`pandas` was extremely useful in helping to produce preliminary quantitative results. Our preliminary results pull from current analysis and past published results (such as Zhao and Fisk 2011). This can be seen in the Table 1.

Table 1 shows the relative ratios of total open magnetic flux and associated parameters for the last three solar minima. The first two rows are the results of Zhao and Fisk 2011. The third row shows the new results of this work from 2024 that further validated the 2011 conservation theory. What is new to this project as a preliminary result is the use of Pandas to study the non-streamer stock region solid angle measurements in conjunction with the measurements of open magnetic flux in the third column. From this preliminary Pandas data analysis,

J.R.M. acknowledges support from NSF Early Career grant 2237435.

TABLE I
STREAMER AND MAGNETIC FLUX CHARACTERISTICS ACROSS SOLAR MINIMA

Solar Minimum	Streamer Half-width (deg)	Non-streamer Stalk Region Solid Angle	Relative Ratio of $\|\mathrm{Br}\|r^2$	Relative Ratio of TOMF
22–23 minimum	25	1.00	1.00	1.00
23–24 minimum	10	1.43	0.74	$\approx$1
24–25 minimum	20	1.14	0.87	0.99$\approx$1

we can see that, for a given minimum, if there is a bigger solid angle of the open flux/non-streamer stock region, physics tells us that the open flux is more spread out. This makes it harder to form what are known as polar coronal holes, which in turn causes a weaker solar cycle. This was easier to visualize through the use of `matplotlib`, which allowed us to create the Figure 1.

First, let us focus on the last panel labeled "Sunspot Number." This is a good measure of the sinusoidal behavior of the solar cycle. For each of the solid angles produced by Pandas and shown in the above table, we can see the aforementioned relationship appear. A bigger solid angle at a corresponding minimum tends to be followed by a smaller peak in the following sunspot number maximum. The opposite is also true. We can also see similar patterns produced by in situ measurements of the open magnetic flux.

The fact that these two parameters alone indicate some sort of accurate predictive model is very promising. Here, we will test decision tree modeling (using CART, random forests, and boosting models), which are commonly used in machine learning modeling in solar physics due to their robustness. We will us generalized additive modeling as well as ridge regression. Most excitingly, we will build and test a simple neural network. The end goal of this project was to see if machine learning or neural networks could help in modeling total open magnetic flux and which models do this task best.

II. METHOD

A. Data Processing

We began our analysis by first performing data processing. In this project, we used data measured in situ by the Advanced Composition Explorer (ACE) [2]. In the context of data collected by spacecraft, "in situ" means data taken at the physical location of the spacecraft at a given moment. This is important because data taken of a particular quantity at one astronomical unit (AU) away from the Sun, for example, may differ when the same quantity is measured at 10 AU away from the Sun. In other words, knowing that a measurement is in situ lends context to further interpretations.

We began our processing of the ACE data by first setting up a filter. We wanted to use ACE data from its full lifetime, which ranges from circa 1998 to the present day. However, portions of the ACE data is tainted after 2011. In 2011, a space weather event irreparably damaged portions of the ACE spacecraft. As a result, certain aspects of ACE data after 2011 are too noisy to properly filter. We can remove these sections and use the rest of the data post-2011, however.

The filter was derived from applying `pandas` filtering to data taken from the ACE Solar Wind Ion Composition Spectrometer (SWICS) instrument [3]. Specifically, we use time-series data describing various charge state ratios of certain elements, such as Oxygen-7+/6+ and Carbon 6+/5+. Why these and other parameters are used to apply a filter is beyond the scope of this paper. This data was provided to us as an `IDL .save` file, but we given that `Python` was the language used for the project, we used the functionality of `scipy` to load the file into a `pandas` dataframe.

With the filter set, we loaded two of the main datasets used for analysis into a `pandas` dataframe and combined them in a way to calculate the total open magnetic flux (our desired quantity of study) at one day intervals across a 24-year time range. We included another quantity called "sunspot number" as an extra predictive parameter through the use of a package called `hapiclient`. In solar physics, two of the biggest problems in conducting research is finding data and accessing it. The `hapiclient` package is one of the newest solutions to the latter problem, as it provides a streamlined way to load data from an online database and load it into a `pandas` dataframe inline in a program.

In the end, we ended the data processing step by splitting and scaling our data into features and observation dataframes. Because we wanted to study the total open magnetic flux, we stored this in an observation column by itself. Our features were observations of the magnetic field in both the RTN and GSE coordinate systems, the RMS values of underlying high-resolution measurements, the variance of the absolute value of the total magnetic field, the aforementioned charge state ratios, the solar wind speed, sunspot numbers, and date. Using `sklearn`, we further split the data into training and validation sets, as well as tensors and data loaders for the neural networks.

B. Machine Learning Modeling

1) Decision Trees: Decision trees have been a common method used to apply machine learning to solar physics. They are highly interpretable and versatile. More importantly for our purposes here, decision trees excel at capturing the behavior of non-linear relationships. Everything about the scenario we are modeling here is non-linear, from our response variable (the total open magnetic flux) to the very nature of the solar cycle. Here, we model three different types of decision tree models and measure their effectiveness.

We first modeled using Classification and Regression Trees (CART). Like all decision tree models, CART builds a tree consisting of nodes made by decision points, which in turn hold a predicted value for our response variables. To determine

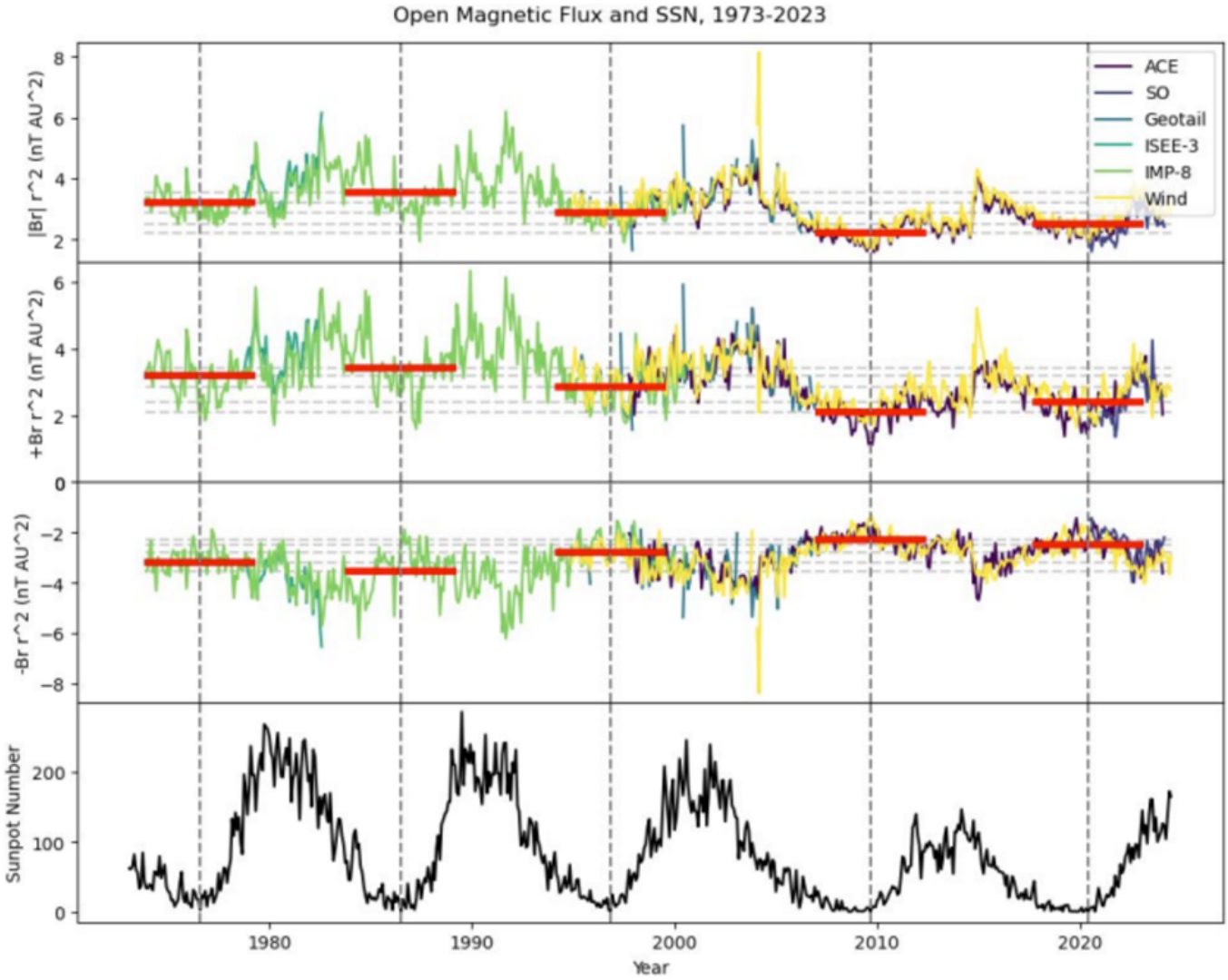

Fig. 1. Open magnetic flux 1973-2022 measured using monthly-averaged magnetic field and ephemeris data from ACE, Solar Orbiter (SO), Geotail, ISEE-3, IMP-8, and Wind. The vertical dashed lines represent solar minima. The horizontal dashed lines and red lines indicate the position of the average open magnetic flux across a given minimum.

how to split the data at a node, CART uses residual reduction in a ”greedy” manner. The lower the residual reduction, the better the fit. We ”prune” the tree, or reduce overfitting, by adjusting the `max_depth` parameter. Here, we determined that setting this parameter to four was the most appropriate way to prune the tree.

Next, we modeled using Random Forest Regression. This modeling method implements bootstrap sampling to create random subsets of data to fit to the data and predict the response. It is particularly useful when predicting continuous variables, which the total open magnetic flux happens to be. Random forest regression also excels over many other models in handling non-linearity. Finally, this model is most useful to us because it works well with large datasets. Here, our feature variable had dimensions of approximately 15 by 5200 and our response variable also had 5200 points. We found that using 100 estimators resulted in the best fit to our data.

Finally, we modeled using boosting decision trees. On their own, decision trees are weak learners, but boosting decision trees combine multiple weak decision trees to create a strong learning model. Boosting decision trees are strong because of their high accuracy, while still being robust and flexible. However, they are not infallible. We had to be very careful in tuning our boosting decision tree model because of how prone they are to overfitting. We found that using 100 estimators and a learning rate of 0.1 fit our data the best without overfitting.

All three of the best decision trees were modeled/visualized using `matplotlib.pyplot`.

2) Generalized Additive Models (GAMs): In addition to decision trees, we also wanted to model using GAMs. Because of the large feature dataset and set target response variable, we deemed that GAMs would be appropriate to use because of

their flexible approach to modeling non-linear relationships, like we have here. GAMs allow for one to model the relationship between the independent variables and the target by summing separate smooth functions for each predictor. This creates a highly interpretable model that can capture non-linear effects without assuming complex interactions unless explicitly modeled.

Because we had 15 features, we composed a linear GAM composed of 15 smoothing splines. Smoothing splines were determined, at least on this first pass, to be the best for the levels of smoothing needed to properly fit the data. We quantitatively supplemented these observations by implementing a grid search to help in finding the optimal level of smoothing for each feature via cross-validation, automating hyperparameter tuning.

After fitting the data to the optimized model, we plotted the partial dependencies of each feature using `matplotlib.pyplot`. GAMs are highly interpretable, allowing us to assess the effect of each feature individually.

3) Linear Modeling - Ridge Regression: Because of he non-linear makeup of our data, we recognize that linear modeling is not ideal for our purposes here. However, we wanted to prove this. We decided that the best type of linear model would be ridge regression because it can be used to mitigate multicollinearity, reducing instability. It does this by implementing a regularization term that helps do this, we we designated as alpha. To choose the best value for alpha, we implemented ridge regression cross-validation across a wide range of possible alpha values, finally settling on an approximate alpha of 33.93.

C. Neural Network Modeling

We built a small neural network from scratch to implement here. This required two basic functions: one function to train the model and another to build it.

1) Building the Training Function: The training function takes in the type of model desired, the training and validation data, the desired loss function, an optimizer, the desired number of training epochs, and a patience value for early stopping.

Throughout the training, it was important to keep track of the training and validation losses, a task best suited to lists. These were initialized first and returned for use later.

We designed the model to train over epochs. Because the neural network training is complex, we used `for` loops to train individually over each epoch. There might have been a way to vectorize and speed up the use of this function, but we decided that this would add unnecessary complexity. Within each epoch, we zeroed out the gradient at the start and generated the loss using the loss function applied to the model outputs and true values. We computed the gradients of this loss with respect to the model parameters (this is called a backward pass) and took a step with the optimizer to update model parameters using computed gradients. This allowed us to retrieve the training and validation losses at each epoch.

We also implemented a patience aspect to the function to help with training time. If the losses do not improve within a range of epochs set by the patience value, the loops are broken and the training stops early. Ideally, this means that the model will not reasonably improve with any more training, reducing the already expensive computation time.

2) Building the Building Function: The function to actually build the model is significantly simpler than the training function. It takes in how many hidden layers and neurons the network will create, as well as choosing an activation function from a preset activation map. Here, we allow for the use of the following activation functions:

- **Rectified Linear Unit (ReLU):**

$$f(x) = \max(0, x) \tag{1}$$

- **Hyperbolic Tangent (tanh):**

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

- **Sigmoid:**

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

- **Leaky ReLU:**

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{4}$$

where α is a small constant, typically $\alpha = 0.01$.

We felt that these were best suited for modeling the non-linear behavior of the TOMF and associated parameters. We also included dropout in our build function. Dropout randomly selects some neurons to be set to zero when training the model. This helps the model correct mistakes from previous hidden layers and improves the performance in fitting the data.

After these parameters were defined, we were able to actually build the model layer by layer. This sequence of layers was marked to be returned. This return was then later fed into the training function to fit to the data.

3) Optimizing the Model: In actually running the model, there are a plethora of parameters to fine tune and optimize. Of the parameters, we felt that the number of hidden layers, number of neurons, specific activation function used, and whether dropout was implemented were the most important. However, to search through a wide variety of all of these parameters in different combinations would be extremely time and computationally expensive. Training even simple neural networks like the one we built here is already expensive in this manner, so we needed to adapt accordingly.

We decided to implement Bayesian optimization with the `optuna` package. This method is efficient at exploring parameter space by focusing on promising regions, adapting search based on past performance. It is also ideal here because computational resources are moderate, and we want to look for smarter exploration based on empirical feedback. `optuna` allowed us to suggest multiple neurons, hidden layers, each possible activation function, and the dropout condition be used in different combinations without spending to much time.

III. Results

A. Machine Learning Modeling

1) Decision Trees: The results of the CART decision tree can be seen in Figure 2. Visually, the orange colored box/node is what the eye is drawn to first. Clearly, it is an outlier and can be thought of as a special case. Looking further into the tree, the x-component of the magnetic field in GSE space is the most used feature. It appears in several branches, suggesting it's highly predictive. The RMS values of underlying high-resolution measurements and solar wind speed appear deeper in the tree, refining predictions once larger splits are made.

From these and other observations we can see some key takeaways. Low `Bmag` and `Bgse_x` values predict small and negative values for the TOMF, and vice versa. The model tends to have more confident predictions (lower error) where more samples are grouped and less variance exists — mostly in the left subtree.

As stated before, we decided on using 100 estimators for the random forest regression. This is still the best choice. We do not include a plot of the decision tree here because the tree is so overfit that it is impossible to interpret and would add nothing of value by being included in this paper. We modified the regression to a high degree, but 100 estimators was still the best, just incredibly overfit.

The results of the Boosting decision tree modeling can be seen in Figure 3. Once again, `Bmag` is highly influential, and divides the space into two broad behavior regions. The same can also be said for `Bgse_x`, as it acts as a frequent refiner for both left and right sides of the tree. the prediction values are generally small (suggesting it's a weak learner), but a few notable exceptions exist.

Overall, we can see the results of the decision tree models and how they stack up against each other in Table 2.

TABLE II
Comparison of Regression Model Performance

Model	MSE	RMSE	MAE	R^2
CART	0.431274	0.656714	0.495202	0.245697
Random Forest	0.271353	0.520916	0.390083	0.525400
Boosting	0.245968	0.495952	0.379697	0.569799

We can see that CART modeling performed the worst on all metrics. This tells us that it ts limited depth and lack of ensemble power likely lead to underfitting. The random forest modeling offers a strong improvement over CART modeling, likely due to its ensemble of many decision trees reducing variance and improving generalization. However, it is still not quite as good as boosting in capturing subtle patterns. Boosting performed the best out of all three decision tree models. It iteratively corrected previous errors, which helped reduced both bias and variance. However, it only had a slight edge over Random Forest on all metrics—especially in R^2 and MAE.

In summary, ensemble methods (Random Forest and Boosting) clearly outperform a single decision tree (CART) by a significant margin. Boosting is the top performer in this task, achieving the best balance of error minimization and variance explanation.

2) Generalized Additive Modeling (GAMs): The results of the GAM modeling can be seen in the partial dependency plot of Figure 4. The plot shows that the strongest influencers were the tangential, normal, and all three GSE magnetic field components, the magnetic field magnitude, RMS of magnetic field variations, and magnetic field variance.

The tangential magnetic field plot shows a nonlinear, positively increasing effect on TOMF, especially at higher values. This suggests that stronger magnetic fields contribute to higher TOMF. the normal magnetic field plot shows that it is negatively correlated with TOMF, which implies that when the northward component dominates, the TOMF decreases. Magnetic field magnitude appears to be strongly positively related to TOMF, with a curve similar to the tangential magnetic field. This indicates that this parameter is a major driver of the TOMF. The RMS of magnetic field variations appears to have a U-shaped relationship with TOMF. Both very low and very high values result in higher TOMF, with a dip in between. All magnetic GSE components show varying degrees of nonlinear impact, particularly `Bgse_x` which has a valley-like shape, suggesting an optimal range for minimum TOMF, outside of which TOMF rises. The variance in the magnetic field shows a bell-shaped influence; TOMF is maximized around a mid-range value of variance.

The sunspot number has less of an impact. It is is generally positively associated with TOMF at high values—likely due to more solar activity. This supports the known correlation between sunspot numbers and solar magnetic activity.

The charge state ratio plots are useful because the ratios themselves are proxies for coronal temperature and plasma source regions. Both exhibit oscillatory, multi-peak structures in their plots. These ratios typically increase with coronal temperature, so higher ratios often correspond to slow solar wind from hotter regions (*e.g.*, near active regions) and lower ratios often indicate fast wind from cooler coronal holes. The model appears to pick up nonlinear relationships between coronal source temperatures and TOMF. Intermediate values seem to maximize TOMF contribution, perhaps due to transition zones (e.g., stream interfaces) being more dynamically structured.

The solar wind speed (h_v) plot is also still useful. It shows non-monotonic, oscillatory behavior, with both positive and negative contributions across different wind speeds. Fast solar wind is associated with open magnetic field lines from coronal holes, which increases TOMF. Slow solar wind can be more closed-field dominated, or associated with more complex topology. The GAM captures complex, nonlinear effects of wind speed on TOMF. Possibly, only certain wind speeds correspond to efficient open flux transport, while others are associated with reconnection and closure. The dips and peaks could reflect interactions at heliospheric current sheets, CME-driven flows, or sector boundaries.

To summarize the results of the GAM, the model captures nonlinear and periodic behaviors effectively—key in space weather and solar physics. Several features (like magnetic field

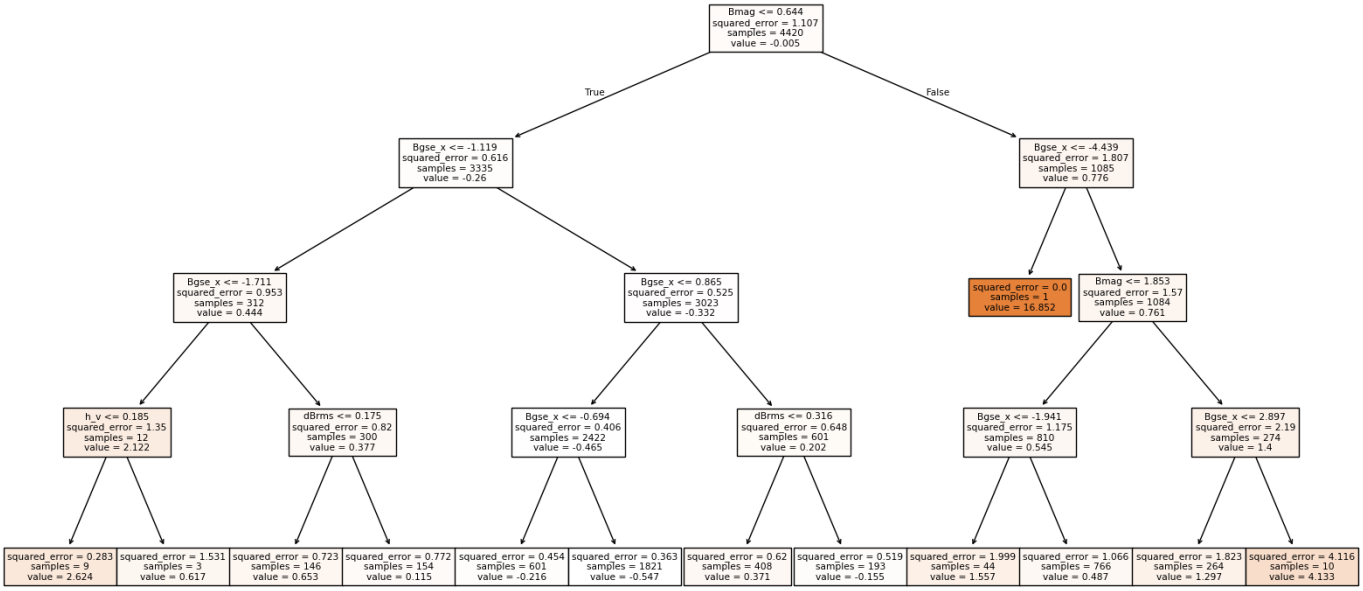

Fig. 2. Results of fitting the TOMF data with CART decision trees and a max depth of four.

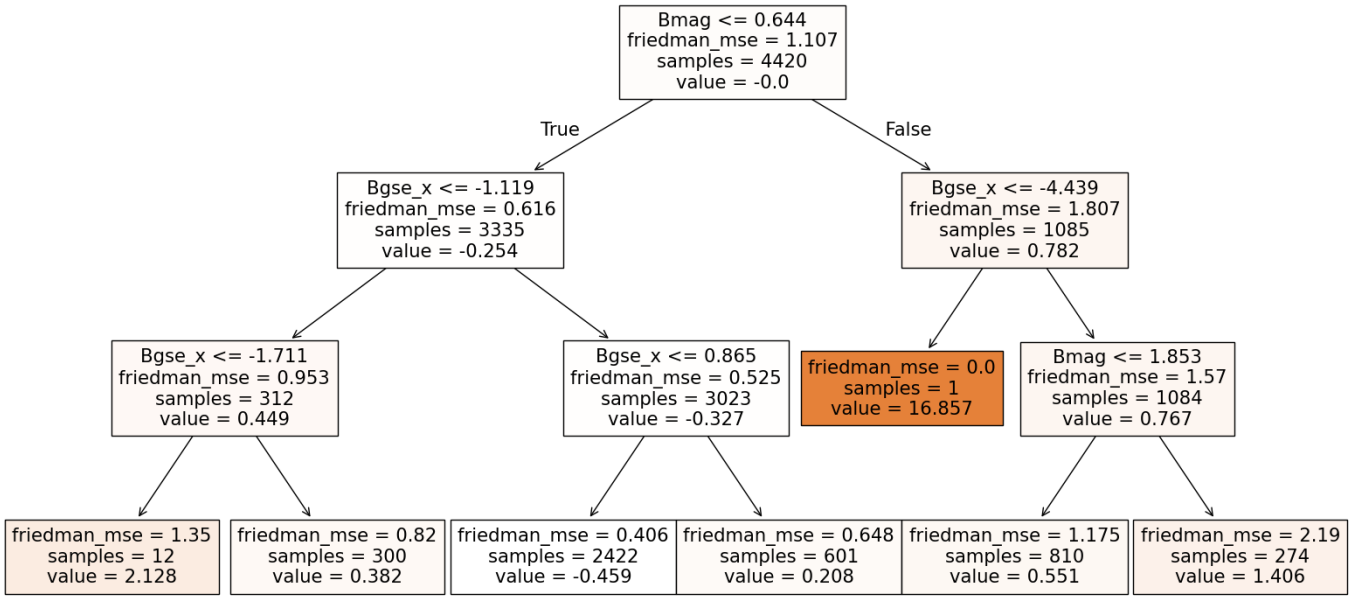

Fig. 3. Results of fitting the TOMF data with Boosting decision trees, 100 estimators, and a learning rate of 0.1.

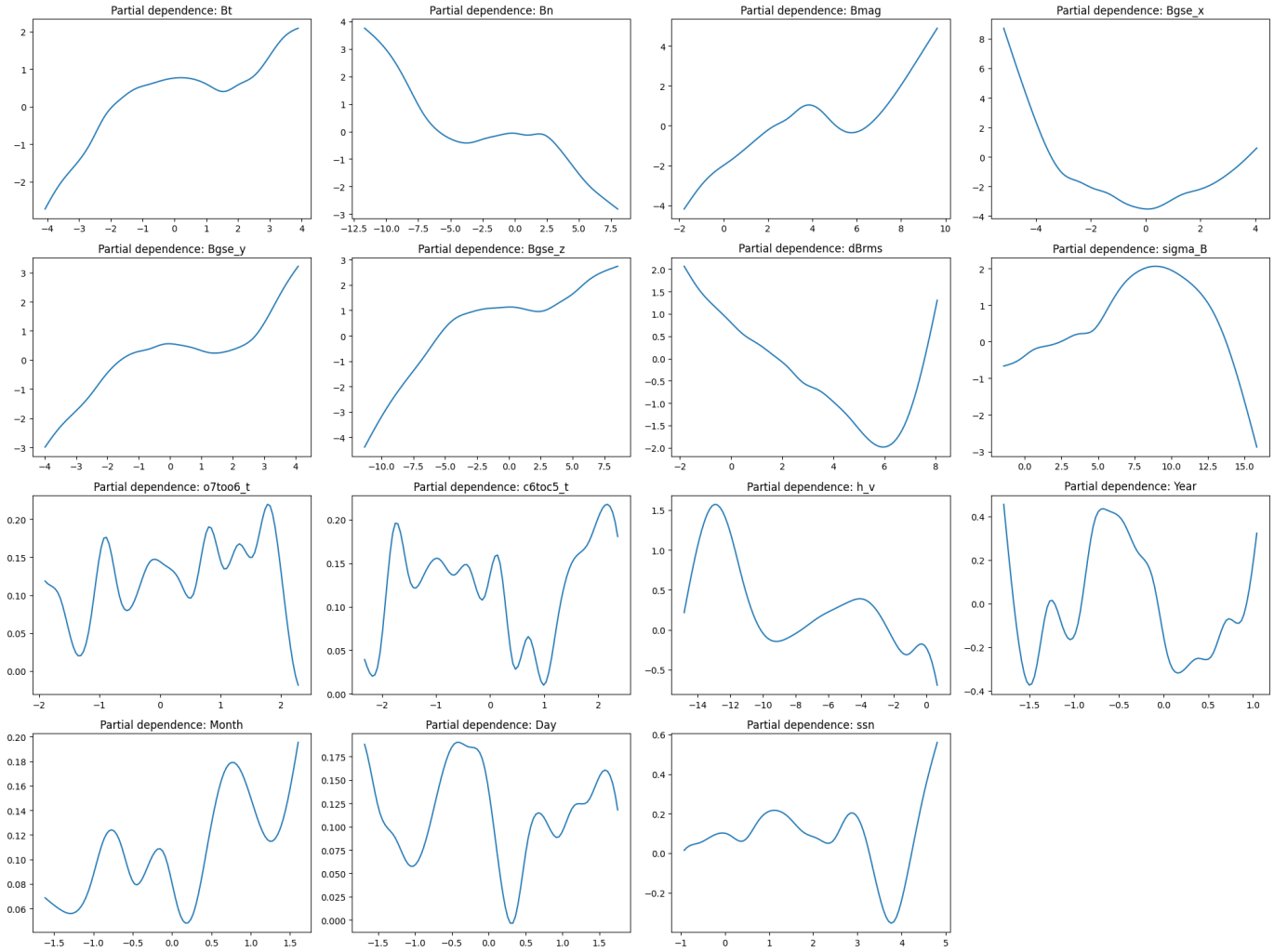

Fig. 4. Partial dependencies of each parameter used to fit the TOMF data.

components and temporal indicators) exhibit clear patterns, highlighting their predictive value for TOMF. Features with flatter curves (or small vertical range) likely have weaker influence in the model.

3) Ridge Regression: The final machine learning result comes from ridge regression. The only real interpretable result that the model could create was the mean squared error. This lack of robust interpretable results automatically makes it the worst of the machine learning models used here. We report an MSE of approximately 0.47, even worse than the worst of the decision tree models. This is not surprising, as a linear model can not be expected to work well with clearly nonlinear data, but including it here was a good exercise, regardless.

B. Neural Network Modeling

The `optuna` code found that the best parameter combination for our neural network was having eight hidden layers, 53 neurons, the hyperbolic tangent activation function, and ignoring dropout. This resulted in a training loss of 1.0986 and a validation loss of 0.587. This can be seen in Figure 5.

These results are obviously not optimal. After reviewing the code used to make the model, we have determined that the issue is likely not a coding error, especially since we implemented a robust approach to optimize our parameter choice.

We instead look at the data itself for why the results here are poor. the data might be noisy, which neural networks seem susceptible to. High levels of noise or errors in the data can mislead the model during training, making it difficult to learn meaningful patterns. The features used could also not be perfect for predicting TOMF, which would cause the model to learn poorly. This seems more likely, especially given the relatively high prediction errors found from the machine learning models.

IV. Conclusion

Our analysis demonstrated that gradient boosting regression using decision trees, with a mean squared error of 0.245, outperformed other methods due to its ability to capture complex nonlinearities in the TOMF data. Gradient boosting

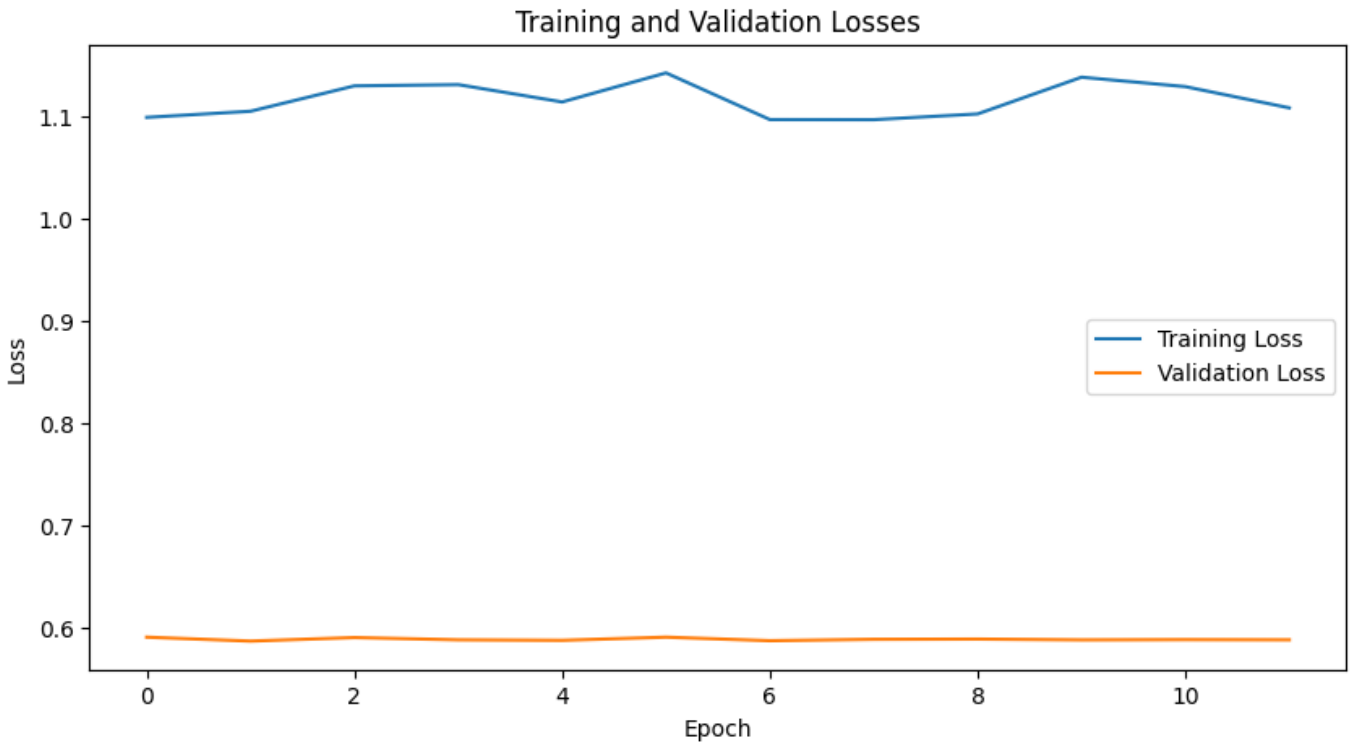

Fig. 5. Training and validation loss results of the simple neural network using eight hidden layers, 53 neurons, the hyperbolic tangent activation function, and ignoring dropout.

is extremely robust for these purposes and decision trees are often used in solar physics machine learning modeling, so it is no surprise that this model performed particularly well. While there were no particular insights gained into the temporal behavior of total open magnetic flux itself, we were able to identify through generalized additive modeling that some parameters over others (such as magnetic field magnitude) are well-suited for predictive purposes when it comes to the TOMF.

These results suggest potential improvements in predictive capabilities for magnetic flux, offering more reliable forecasts for space weather phenomena. The implemented models could enhance early warning systems for geomagnetic storms, thereby protecting critical infrastructure. Obviously, there is a lot more to improve on before these aspects can be implemented, let alone relied on, but this is a promising first step in the right direction.

While the neural networks provided robust predictions, they required significant computational resources and large datasets to avoid overfitting. In the future, we may want to examine a wider parameter space with a more powerful computer to see if our neural network modeling can be improved. However, the study's results are limited by the available resolution of the input data, suggesting that higher-quality datasets could improve the predictive performance further.

In future work, we will endeavor to improve the quality of our data. It may also be useful to expand or refine our feature variable space to see what best improves the predictive nature of the TOMF. Here, we used a small amount of simple models, but it would be interesting to expand the volume of models used, as well as seeing if more complex models can offer us any additional insights.

Overall, this study highlights the potential of machine learning and neural network models to contribute significantly to the understanding and forecasting of solar magnetic phenomena, particularly when it comes to open magnetic flux, paving the way for improved space weather resilience.

References

[1] L. Zhao and L. Fisk, "Understanding the behavior of the heliospheric magnetic field and the solar wind during the unusual solar minimum between cycles 23 and 24," *Solar Physics*, vol. 274, pp. 379–397, 2011.

[2] M. C. Chiu *et al.*, "ACE spacecraft," *Space Science Reviews*, vol. 86, pp. 257–284, 1998.

[3] G. Gloeckler *et al.*, "Investigation of the composition of solar and interstellar matter using solar wind and pickup ion measurements with SWICS and SWIMS on the ACE spacecraft," *The Advanced Composition Explorer Mission*, pp. 497–539, 1998.