Developing a Data Analysis Tool to Predict Radar Data from Magnetometer Data during the Farley Buneman Instability

By Alan Michael and Dr. Roger Varney – Department of Atmospheric and Oceanic Sciences, University of California Los Angeles

Abstract

During the Farley Buneman instability, large amounts of current are flowing in the ionosphere. Despite being mainly observed through radar campaigns, these currents generate large magnetic fields which should be noticeable on surface magnetometers. By developing a data analysis program using machine learning, we can measure the relationship between radar and magnetometer data and expand the overall dataset of observed accounts of the Farley Buneman instability.

The architecture of this model will incorporate concepts from atmospheric and space sciences as well as machine learning methods like polynomial regressors and support vector machines. This multi-stage model will be used to accurately map incoming magnetometer data to the corresponding radar data. Additionally, the model will predict particle velocities and temperatures to be able to expand the dataset of radar stations using readily available magnetometer data. Additionally, to achieve higher accuracies, hyperparameters will be tuned and set using relevant ionospheric plasma physics concepts.

This experiment can open new research possibilities by being able to better utilize existing magnetometer data to find historical accounts of the FB instability and through predicting radar data from magnetometer data we can expand the working dataset we have for magnetometer stations around the world.

Background:

Due to the large electron velocities in the E-region ionosphere from the Farley Buneman (FB) instability, intuition tells us that there should be a respective large field strength detected by surface magnetometers. But the exact relationship between this data is unknown. Using complex data analysis we were able to verify this correlation and develop a machine learning algorithm using random forest architecture to both predict and classify incoming magnetometer data by whether the FB instability occurs. Additionally, our analysis developed a dataset for Farley Buneman instability occurrences in radar data at the Resolute Bay station and an overall model architecture for analyzing alternating code radar and magnetometer data.

From a space weather perspective, being able to utilize readily available magnetometer data to analyze radar phenomena can vastly improve our understanding of future E-region ionospheric studies. Additionally, sampling bias and the non-random nature of radar campaigns can impede our ability to measure the frequency of FB and other E-region phenomena.

Objectives

Design and Develop a **Machine Learning Model** using a **Random Forest Classifier** that can recognize/predict Farley Buneman instability in incoming Magnetometer Data with > 80% accuracy ..... (1)

Implement a **Software Dataset** that stores notable dates and times of Farley Buneman instability recorded at the Resolute Bay Radar Station ..... (2)

Data Sources

The RISR-N radar at Resolute Bay, Canada provides data in 6 fields:

- Altitudes
- Electron Densities
- Ion Mass
- Electron Temperatures
- Electron Velocities
- E field strength

The RISR-N radar data is generated in the format of h5 files that store data in various tables for accessing. From here, scripts were written to convert the data into usable formats for analysis, predominantly pandas dataframe and numpy arrays. (Figures 1 and 2)

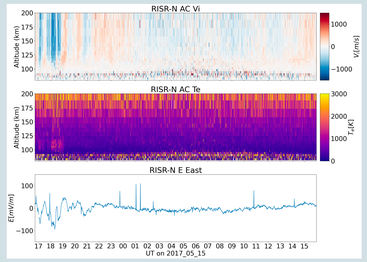

Figure 1: Sample data extracted from the RISR-N radar station's AMISR website.

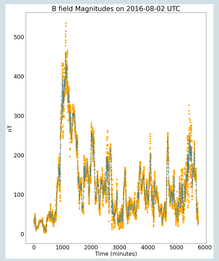

Figure 2: Plot of Magnetometer Data extracted from RES station with trendline

Development of Dataset

In order to run our Random Forest classifier, we needed more data to train on. However, due to the size of the radar data, manually parsing month by month is not a feasible strategy so we developed this data pipeline. The last stage data was manually reviewed and 28 examples of Farley Buneman instability from 2009-2022 were identified and processed.

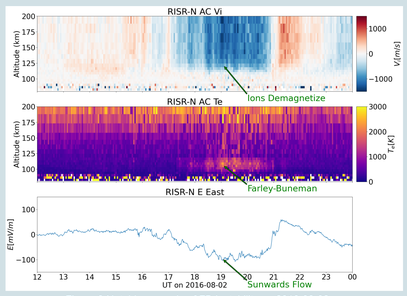

Figure 3 Notable example of FB Instability on 2016-08-02.

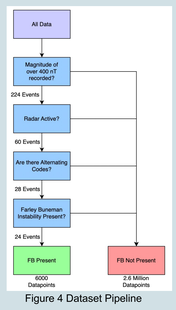

Figure 4 Dataset Pipeline

At this point our data was labelled on the basis of FB Present = 1, and FB Not Present = 0. These labels were manually generated and added into the wider dataset. We have 4 categories of data.

(1) Magnetometer < 400 nT,

(2) Magnetometer > 400 nT, but radar doesn't have Alternating Codes,

(3) Magnetometer > 400 nT, radar has Alternating Codes, and FB is not identified in the radar,

(4) Magnetometer > 400 nT, radar on in the right mode, and FB is identified in the radar.

With only case 4 being labeled FB Present, and All other cases labeled FB Not Present since we don't have enough information on if FB is present or not.

Pre-Processing Function

Scripts were developed to convert missing and NaN values to 0. Additionally, the different datetimes had to be lined up so for simplicity, the analysis converted everything to Unixtime.

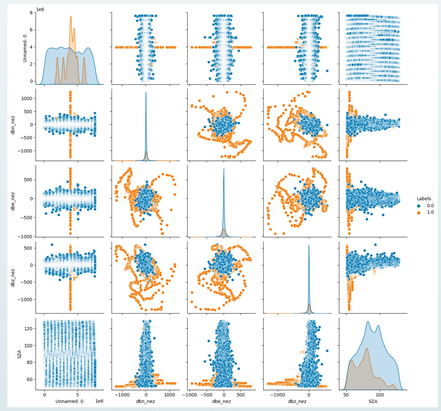

Figure 5: Multivariate Pair Plot of Data Columns

As visible in the plot, the difference between the Blue FB Negative and Orange FB positive samples are clearly statistically significant. With each following different histograms across the diagonals.

Exploring Machine Learning Approaches

Logistic Regression:

With an incredibly high false positive rate and the model struggling to generate accurate predictions for when true positives were presented, the Logistic Regression Machine Learning approach was ruled out as a possibility for the overall design of our architecture (Figure 6).

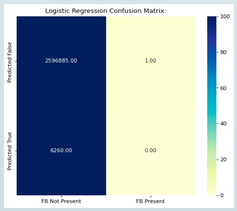

Figure 6: Logistic Regression Confusion Matrix

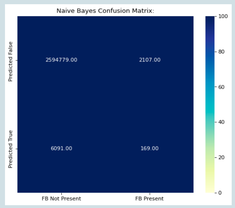

Figure 7: Naïve Bayes Confusion Matrix

K-Nearest-Neighbor Classifier:

So far, the KNN classifier has performed the best. As seen in Figure 8, with a high accuracy, low False Positive rates, and low False Negative rates, this model seems to be the perfect choice. The only downside of this architecture is that each model takes exponentially longer with every additional dataset (Figure 8).

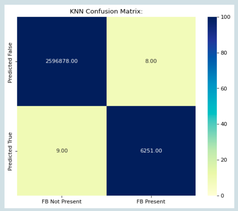

Figure 8: KNN Confusion Matrix

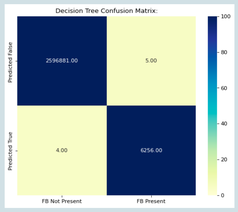

Figure 9: Decision Tree Confusion Matrix

Naive Bayes Classifier:

The model clearly has a hard time discerning the data and this might be attributed to the continuous nature of the data as opposed to more clustered datasets (Figure 7).

Decision Tree:

The Decision Tree Classifier has the best performance from all the models tested. Additionally Decision Trees show a lower False Positive and False Negative rate than KNNs (Figure 8). Overall. The Decision Tree appears to be the most apt approach for classifying incoming magnetometer data (Figure 9).

Bootstrap Aggregation and Ensemble Learning

Since multiple approaches have good performance, we can explore methods of training that utilize various techniques to get higher accuracies. Bagging and Stacking are the most common ways of boosting model accuracy.

Bagging

Bootstrap Aggregation, otherwise known as Bagging, is a technique that boosts the performance of machine learning algorithms by using plurality voting between various simultaneous trained models. We have multiple decision trees that all are trained independently and generate their own predictions and we can then see the majority vote to make a final decision on each given datapoint.

```
{'n_estimators': 160, 'max_features': 'sqrt', 'max_depth': 38}
```

Since we're using multiple decision trees, we have to set the number of estimators and depth of each tree. We trained and developed 10 random forest classifiers with varying depths from 20 to 200. Each tree has 1 of 11 possible max depths from 10-50. Using this, we can determine our optimal parameters.

Hyper-parameter Tuning

Before proceeding to develop the model we first listed our Hyperparameters. Hyperparameters are user-specified parameters that can largely impact the accuracy and functionality of the model. For our approach, our model has 4 hyperparameters

1) Input parameters – List of inputted data columns and size of dataset
2) Training Policy – Loss metric for minimizing per node
3) Tree Max_Depth – Depth of Final Tree
4) Number of estimators - Number of Trees in Random Forest

The training policy decided for this approach is Entropy. At each node, the Tree will try to minimize the entropy of the resulting branches. This was decided to minimize training time.

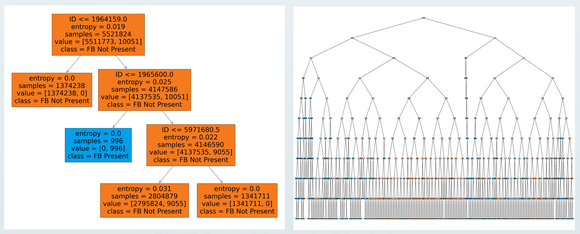

Figure 10: Example of Under and Overfit Decision Trees

With lower depths the tree can generate predictions faster but larger depths have larger accuracies. Due to the volume of magnetometer data, we decided to prioritize fast predictions and testing over marginally higher accuracies.

Development of Random Forest Classifier

The general idea in training most models is developing (1) cost function, (2) backpropagation method, (3) forward propagation, and (4) testing. Our analysis is using a modified fit function to rearrange nodes. Our function works on dividing the datasets into smaller subsets to minimize our training policy (Entropy).

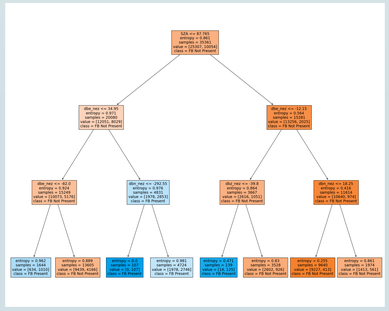

Figure 11: Initial Tree from Random Forest with Blue representing Predicted True (Depth = 3)

After generating sub-trees and picking optimal ones, the tree is now fully trained and fit on the dataset. Figure 11 highlights the architecture of the tree with a bit of meta-analysis visualizing what the tree might be splitting the parameters like.

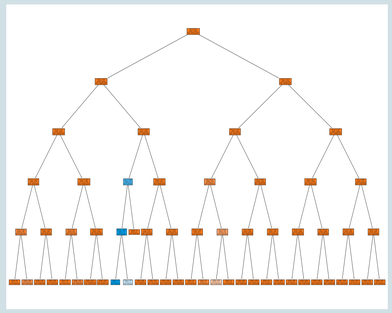

Figure 12: Reworked Tree from Random Forest with Blue representing Predicted True (Depth = 5)

Our training process is very similar to standard Decision Trees except we now have to train multiple decision trees. Since we decided to utilize a depth of 38 and 160 trees, our results should be fairly accurate and not fall into the overfitting issue.

Here are some randomly generated trees for our architecture. At this point we split our data into 70/30 split for training/testing data to not conflate our results.

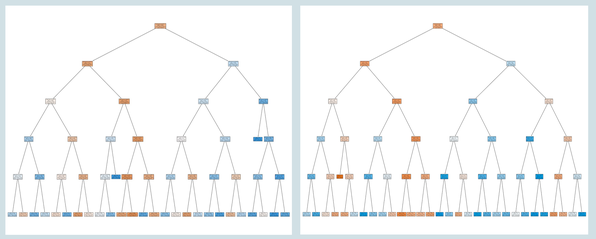

Figure 13: Final Trees from Random Forest Classifier, Blue = Predicted True, Orange = Predicted False

Testing and Rework

Our models' initial phase testing results are good but not approaching the small-scale testing from earlier (Figure 9). This can be attributed to the more refined dataset were using for final stage development additionally, since FB instability is such a rare phenomena the overall performance of the model looks worse just because there are millions of examples of non-FB instability compared to only 32 known instances.

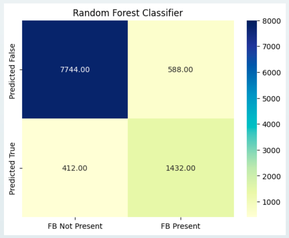

Figure 14: First Stage Testing Results of Model

Metrics:

Accuracy: 0.9981757839438562

AUC: 0.5136790066681996

Precision: 1.0

Recall: 0.9926419866636008

F1-Score: 0.9853914631362702

First Stage Rework

As seen in Figure 14, the AUC is very low due to the rarity of FB instability. To combat this, we decided to reduce the negative samples to be 10% of the initial quantity this would improve the overall specificity of our model and lead to improved performance across the board. With our new dataset developed we proceeded to retest and quantify our model's performance.

metrics	0.0	1.0	accuracy	macro avg	weighted avg
precision	0.988785	0.874870	0.96887	0.841827	0.878764
recall	0.944547	0.930489	0.96887	0.837518	0.918870
f1-score	0.953694	0.949992	0.96887	0.851843	0.938199
support	10820.000000	4336.000000	0.96887	15156.000000	15156.000000

Figure 15: Second Stage Testing Results of Model

Second Stage Rework

Our model's performance varies greatly between true and false samples but still stays above our threshold, so we reworked the hyperparameters to be more manageable with max_depth of 10 and the number of estimators being 100. At this point, we re-evaluated our model's performance.

Metrics:

Accuracy: 0.9981757839438562

Precision: 1.0

AUC: 0.8273945680099016

F1-Score: 0.9853914631362702

Recall: 0.9926419866636008

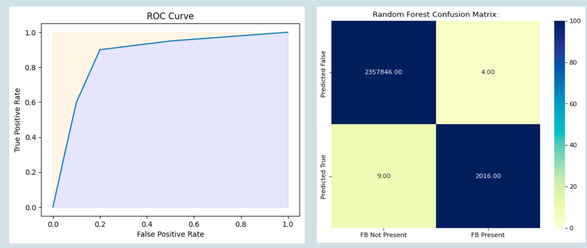

Figure 16 and 17: Final Stage Testing Results for Random Forest Classifier. ROC Curve and Confusion Matrix

Data Analysis

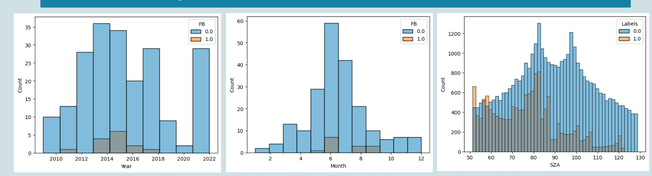

Figure 18: Frequency distributions of FB instabilities with labeled circumstances noted

Frequency distributions were generated to measure all events when the magnetometer recorded over 400 nT and where Farley Buneman instability was specifically labeled and found using the corresponding Radar Data. We found a clear association with the solar cycle (Year Distribution), preference of summer months (Month Distribution), and preference of daytime (SZA Distribution).

Conclusion and Future Work

Despite being a notable phenomena in Ionospheric studies, the scarcity of analysis on Farley Buneman instability using machine learning techniques can be much attributed to the lack of a strong data platform. On this front, we have succeeded by generating a sample dataset from 2009 to 2022 of notable Farley Buneman instability occurrences.

After considering the initial criteria specified in this analysis: our solution was a success. With over 99% accuracy, the Random Forest approach can accurately record when Farley Buneman occurs in magnetometer data (precision) as well as correctly identify when it doesn't (recall). Additionally, clear physics conclusions can be drawn from our data analysis.

Future Work:

- Generative Models for creating Radar Data from Magnetometer Data
- Cross-validation with data from nearby stations to expand dataset
- Meta-analysis on Machine Learning Processes to find formulaic relationships between events

Acknowledgements

This work is supported by the Resolute Bay Observatory which is a major facility funded by the National Science Foundation through cooperative agreement AGS-1840962 to SRI International. RISR-N data is available through the SRI International ISR Database at https://amisr.com/database/