VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Mining Solar Wind Data for Turbulent Microscales

Kevin de Lange

Supervisors: Marcus Frean and Tulasi Parashar

Submitted in partial fulfilment of the requirements for
Master of Computer Science.

Abstract

Studies on turbulence in solar wind are defined on many scales, and phenomena governing large-scale motions are generally thought to be separate from effects governing small-scale behavior. This provides an opportunity for data mining and machine learning to investigate whether, and to what extent, the Taylor microscale can be correlated to large scale flow characteristics. This work takes an exploratory approach, creating a data set and using three machine learning algorithms to discover these connections.

Acknowledgments

I would like to acknowledge my supervisors, Marcus Frean and Tulasi Parashar, for their support and guidance throughout the project and, as always, my family for their continued support throughout my life.

Contents

Figures

Chapter 1

Introduction

Artificial intelligence (AI) and machine learning have greatly advanced research in other areas of science in recent decades; one of such fields is the study of space plasmas in the form of solar wind. With its rapid advancement alongside the increase in computation power available, AI has both enabled automation of otherwise manual processes typically performed tediously by humans, and advanced theory and understanding through applying newly available machine learning tools in various ways.

The theory behind turbulence in solar wind exist on many different scales of both length and time [1]. Additionally, the study of turbulence in solar wind specifically differs from traditional fluid turbulence due to its mostly collisionless nature. Traditional magnetohydrodynamic behavior is observed at large scales, but at smaller, particle scales, the flow enters a kinetic regime [13]; kinetic physics therefore play a large role in solar wind phenomena [14]. For instance, the propagation of turbulent eddies originates from large scale fluctuations, said to be in an energy-containing length scale. Here, a length scale can loosely be considered as the size of a turbulent eddy. These large eddies then break up and transfer their kinetic energy to progressively smaller eddies in a so-called energy cascade, until eddies become so small that viscous effects become significant and dissipate the eddies. Turbulence theory have defined specific scales that characterize this process. Generally, the effects taking place at these scales are considered to act independently; for instance, viscous effects have a negligible impact on large scale flow effects, but becomes dominant at small scales [1] [15]. Theory has been established on these small scales, but these scales are difficult to measure and no definitive data set exists to investigate the nature of these scales. Machine learning and data mining techniques therefore provide a unique opportunity within this study. Concepts in these areas can be applied to the ongoing research that exists in solar wind turbulence in an exploratory manner, with a focus on uncovering the nature of small scale characteristics in relation to other solar wind characteristics.

1.1 Goals and contributions

The nature of this work is exploratory rather than formative, applying techniques from machine learning and data mining concepts to investigate the nature of small-scale turbulence behavior, rather than designing a machine learning model with the express purpose of prediction. The goal of this exploratory work is therefore to harness the large amounts of data collected on solar wind to produce new knowledge towards the theoretical study of the behavior of solar wind turbulence on its various scales. This is done through both analysis of the data set itself and through applying explainability techniques to machine learning models to extract knowledge. The contributions from this work are as follows:

- A survey is done on the current state of machine learning in turbulence physics.
- Multiple data sources are processed and consolidated into a single data set with the purpose of investigating the Taylor microscale of solar wind turbulence and its relationships with global solar wind characteristics.
- Exploratory analysis is done on the created data set which provides insight into apparent connections between the investigated parameters to the Taylor scale.
- The methods of estimating turbulent length scales in literature are implemented and evaluated.
- Three machine learning algorithms are applied to the data set, using different explainability methods to further gain insight into the nature of the Taylor scale with respect to the investigated parameters.

1.2 Report layout

The report begins with a literature review in chapter 2, which surveys necessary background in turbulence physics necessary to put this work into context, literature on current active areas of research in turbulence physics, and literature on the current state of machine learning applications within the field of solar wind studies. Next, chapter 3 provides background to both terms necessary to understand the data collected and the machine learning models used. The data processing pipeline to create the data set is then outlined in chapter 4. Exploratory data analysis is then outlined in chapter 5. Finally, the implementation of and results from machine learning algorithms is discussed in chapter 6, and conclusions are summarized in chapter 7.

Chapter 2

Literature Review

This survey of literature exists in two main parts: one concerned with relevant background and current studies in turbulence studies, specifically turbulence in solar wind, and one concerned with the current state of machine learning in the field of solar wind research.

2.1 Background on turbulence

Though the nature of turbulent flow is quite intuitive, a precise mathematical definition is difficult to formulate; instead, turbulent flow is often defined as a set of characteristics [15].

- Irregularity: referring to the randomness of turbulent flows. As a result, deterministic models of turbulence are not possible and researchers must rely on statistical representations [15].
- Diffusivity: the most important characteristic of turbulence when used in practice. The diffusivity of a flow refers to the increased rates of transfer and mixing of properties like of momentum, heat, mass, and moisture; such a feature has applications in aviation, manufacturing, and many other industries. For instance, the diffusivity of flow over an airfoil helps prevent boundary separation, a phenomenon of laminar flow that decreases lift over an airfoil at high angles of attack. Though laminar flow on an airfoil is more efficient, creating less drag, designers often opt for turbulence generators at the leading edge of a wing, as the diffusive nature of turbulent flow prevents boundary layer separation [15].
- Large Reynolds numbers: turbulent flows are often the result of an unstable laminar flow when the Reynolds number of that flow grows too large. The interactions of this are highly complex and intractable without sophisticated mathematical tools [15].
- Dissipation: viscous losses in the form of internal shear stresses will decrease the kinetic energy of a turbulent fluid. Energy must be supplied to a turbulent flow, otherwise it will eventually dissipate [15].
- Continuum: a continuum phenomenon is still governed by known fluid mechanics, as the scale of turbulent flow are far above the molecular scale [15]. However, this definition does not hold for plasmas at small (particle) scales, where magnetohydrodynamic behavior transitions into that of kinetic plasma [14].
- Flows: turbulence is often spoken of as a characteristic of flows, not of fluids; the behavior of turbulence is not governed by the molecular makeup of the fluid present in a flow [15].

2.2 Active research in turbulence physics

The study of turbulence is complex and defined on a wide range of length and time scales. The research in these length scales is ongoing, especially in low-density plasma settings [16] and further understanding of the nature of such scales and interactions with other properties contributes to a more grand understanding of the mechanisms behind physical phenomena concerning plasma, such as magnetic reconnection [3]. In fact, the different scales in turbulence are considered to act independently of each other; large scale movements are mainly determined by geometrical constraints and boundary conditions, while small scale behavior is determined by energy transfer, molecular interactions and viscosity. The exploration of magnetohydrodynamic scales is therefore a promising avenue for the application of data mining and machine learning.

To illustrate the multi-scale nature of plasma turbulence, one often refers to an "energy cascade" that spans from the largest to the smallest scales; kinetic energy that sustains turbulent eddies is produced at the largest scales, and this energy is transferred successively to smaller and smaller scales, breaking up into smaller eddies [1]. When discussing length scales of turbulent flows, a physical analog is therefore to consider the physical size of a turbulent eddy, which is loosely defined as a turbulent motion within a fluid [1]. When eddies become unstable, they therefore break up into smaller ones, following a cascade of energy to a smaller scale. This cascade continues until eddies become sufficiently small such that viscosity effects become significant enough to dissipate the kinetic energy completely into heat, representing the end of the energy cascade. An overview of the mechanics of the energy cascade are shown in Figure 2.1 and Figure 2.2: kinetic energy enters the system at the largest scales, called the energy-containing range, where energy is cascaded to smaller scales where eddies dissipate due to viscous effects.

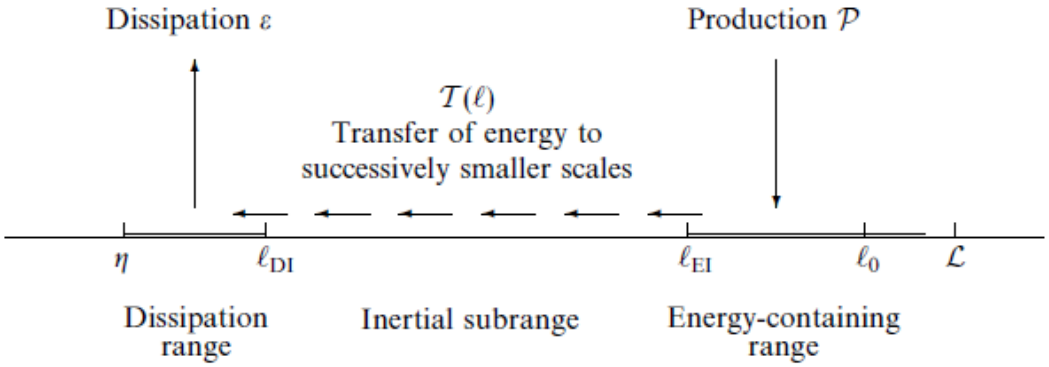


Figure 2.1: Schematic depiction of turbulence energy cascade mechanics over length scales (1) [1]

In these figures, various length scales are shown which have roots in traditional hydrodynamic theory and concepts in plasma physics theory. Of particular interest is the Kolmogorov length scale, η, representing the smallest length scale occurring in the system [1] [17]. Larger length scales are represented in the energy-containing range, where $\mathcal{L}$ is the flow scale, which can be interpreted as an upper bound to eddy sizes, and ℓ_0 is the largest eddy size, naturally smaller than $\mathcal{L}$. Boundaries between the inertial subrange and both the dissipation and energy-containing range are represented by ℓ_{DI} and ℓ_{EI} respectively [1]. The inertial subrange is where all of the dissipation occurs within the energy cascade; in fact, the dissipation in an energy spectrum is shown to follow a power law of the form $E(\kappa) = C\epsilon^{2/3}\kappa^{-5/3}$ [1], where C is a constant shown to be 1.5 through experimentation,

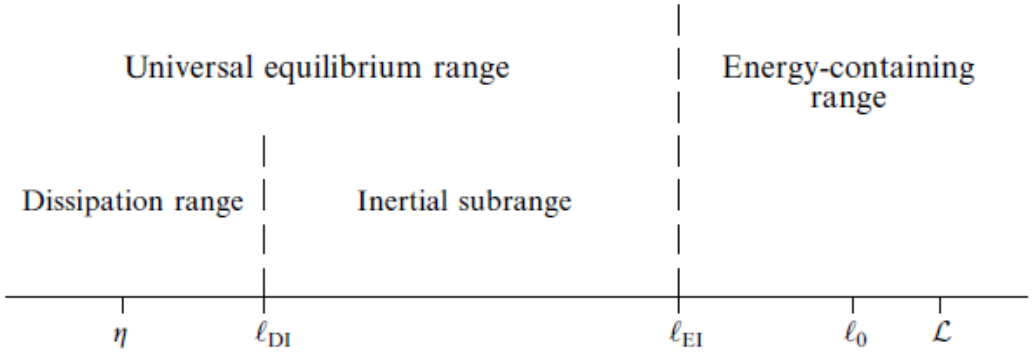


Figure 2.2: Schematic depiction of turbulence energy cascade mechanics over length scales (2) [1]

ϵ is the dissipation rate, κ is the wave number in the power spectrum, and $-\frac{5}{3}$ is known as the Kolmogorov constant. Indeed, with the various scales in turbulence, it is useful to view the energy cascade in a power density spectrum. One typical example of such a spectrum is shown in Figure 2.3 in a power density spectrum of a magnetic field time series. Here, the same principles are visible, energy originates from large scale eddies (low frequency), and is transferred according to a $-\frac{5}{3}$ power law until dissipative/viscous effects become significant.

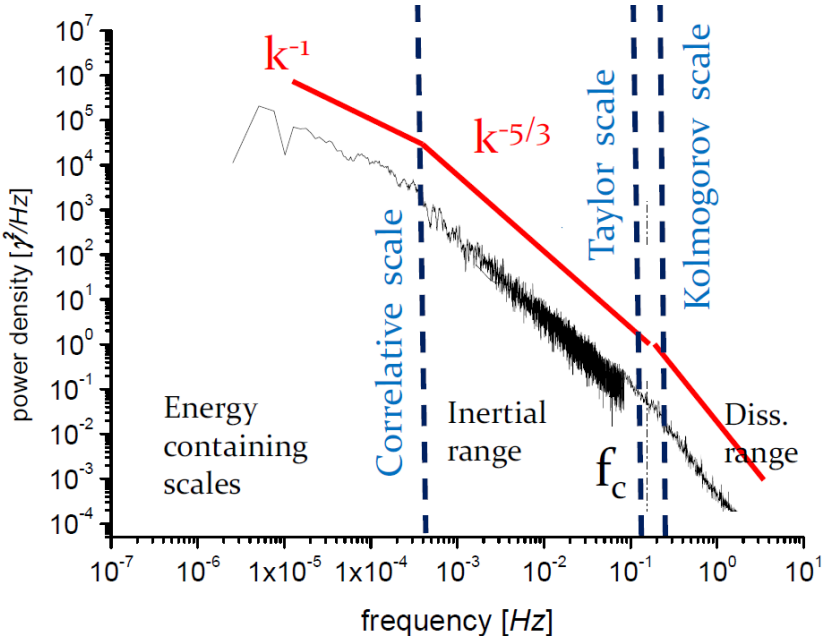


Figure 2.3: Magnetic field power density spectrum with relevant length scales annotated: energy originates from large eddies in the energy-containing range, which then cascades its energy to smaller scales, until dissipation effects come into effect [2]

Here, the Kolmogorov scale is once again visible, and two more length scales are introduced: the correlation scale and Taylor scale. The correlation scale λ_c, also called the integral scale, represents the largest eddy sizes, comparable to ℓ_0 in Figure 2.1 and Figure 2.2, or more concretely the separation for which eddies remain correlated [2] [15]. The Taylor scale, also called the Taylor microscale, being among the smallest length scales, is related

to dissipation, namely where dissipation begins to take effect. It can be interpreted as the largest eddy sizes before dissipative effects take place, or the smallest eddy size within the inertial range [2] [15] [3]. The Taylor scale is distinguished from the Kolmogorov scale in that the latter is defined as when eddies become critically damped, rather than when this dampening begins; it follows that the Taylor scale is larger than the Kolmogorov scale in classical hydrodynamics [3].

Such scales can be determined through analysis of the autocorrelation functions of magnetic field data [2] [15] [3]. This autocorrelation function is defined as Equation 2.1 [3].

$$R(r) = \langle \mathbf{b}(x) \cdot \mathbf{b}(x+r) \rangle \tag{2.1}$$

Here, $\mathbf{b}$ is a three-dimensional magnetic field time series, and r is the specified lag, and the $\langle \cdot \rangle$ function denotes an autocorrelation of the time series at the specified lag; such a function computes the correlation of a time series with itself when shifted by a given offset. Both the correlation scale and the Taylor scale can be determined through calculating the autocorrelation of a magnetic field time series; a sketch of such a function is shown in Figure 2.4. When normalized, autocorrelation functions begin at $f(0) = 1$.

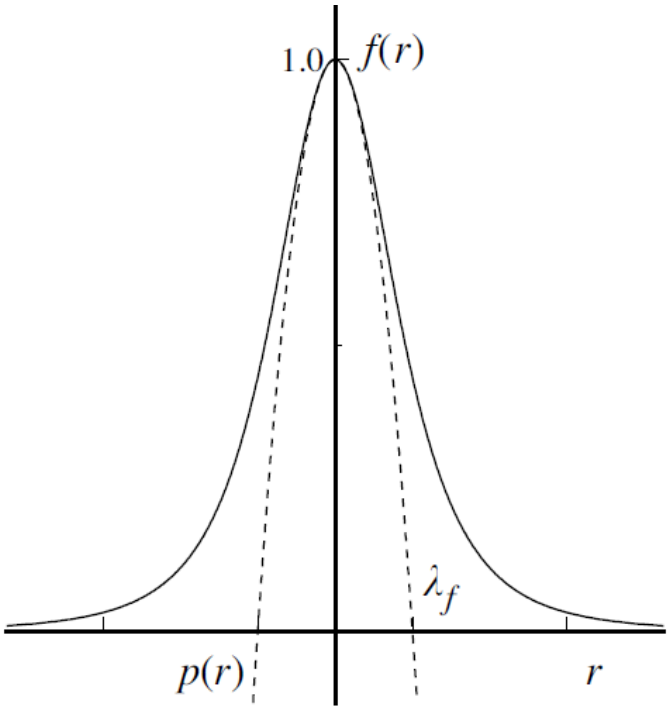


Figure 2.4: Sketch of a typical shape of a magnetic field autocorrelation function [1]

The transverse Taylor scale can be determined with respect to a parabolic fit near the origin of the autocorrelation function. Two equivalent interpretations exist:

- The curvature of a parabolic fit at the origin
- The x-axis intercept of a parabolic fit

The second interpretation results from a Taylor series representation of a parabolic fit, as shown in Equation 2.2 [1]. As shown in Figure 2.4, a typical autocorrelation function begins at $f(0) = 1$, and is symmetrical; it follows from symmetry that $f''(0) = 0$, resulting in Equation 2.3[1]. The Taylor microscale is then defined as the value of x where $f(x) = 0$, leading to the expression in Equation 2.4[1]; the transverse Taylor microscale is then defined

by Equation 2.5[1]. Equivalently, the transverse Taylor microscale can be defined in terms of the curvature (second derivative) as in Equation 2.6 [3].

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 \tag{2.2}$$

$$f(x) = 1 + 0 + \frac{1}{2}f''(x_0)(x - x_0)^2 \tag{2.3}$$

$$x_{f(x)=0} = \lambda_f = \left(-\frac{f''(0)}{2}\right)^{-1/2} \tag{2.4}$$

$$\lambda_T = \frac{\lambda_f}{\sqrt{2}} \tag{2.5}$$

$$\lambda_T^2 = \frac{f(0)}{f''(0)} \tag{2.6}$$

As Taylor scale estimation is related to a parabolic fit near the origin, one requires sufficient temporal resolution in measurements [3]. A concrete example is shown in Figure 2.5 [3]; while the overall shape of the autocorrelation resembles an exponential decay, a zoomed in slice near the origin as seen in the inset plot shows a locally parabolic shape. Here, the parabolic fit is shown as the black curve, which quickly drops to zero. A sketch of this effect is shown in Figure 2.6

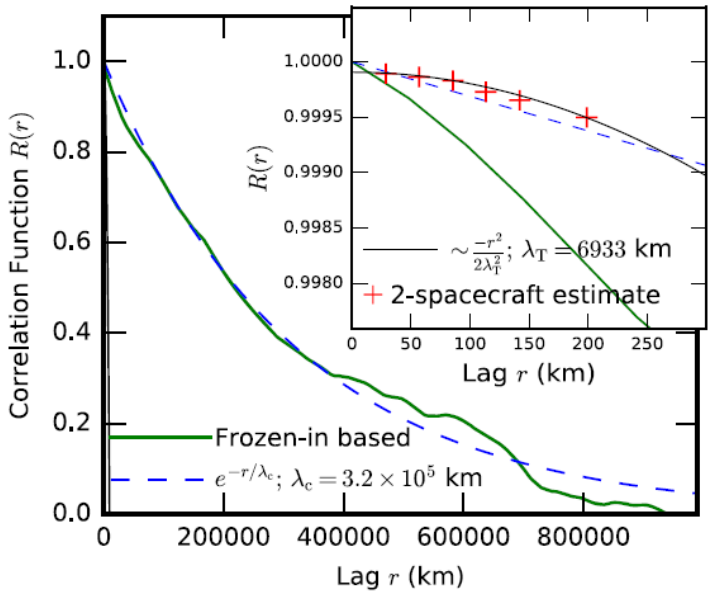


Figure 2.5: Computation example of autocorrelation function from a study on MMS spacecraft data [3]

Figure 2.5 also shows the method of calculating the correlation scale; an estimation can be made through an exponential fit of a larger portion of the autocorrelation function of the form e^{-r/λ_c}. Using any method of best-fit finding, the resulting λ_c value is then the correlation scale. In Figure 2.5, lags are expressed in units of length (km), while autocorrelation functions are originally in terms of lags; lags can be converted to either time or length units by multiplying the number of lags by the data sampling frequency to arrive at time units, then by flow speed to arrive at length units; this is possible through following Taylor's

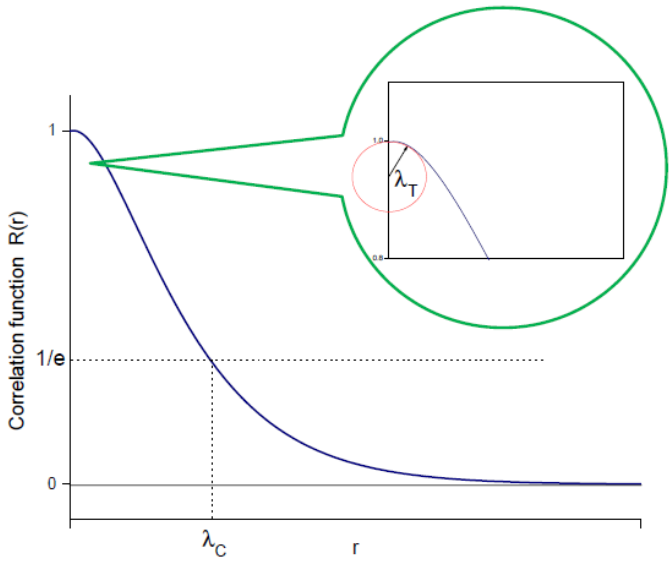


Figure 2.6: Sketch of a typical autocorrelation function, showing the overall exponential shape of the curve and the local parabolic behavior near the origin [2]

frozen-in hypothesis, which allows the interpretation of temporal lags as spacial lags [18]. Taylor's hypothesis is generally applied when flow speeds are sufficiently fast such that any nonlinear changes of the flow is negligible throughout the lag time [18]. An alternative "trick" often used is to forego an exponential fit and simply find the point at which the autocorrelation function takes the value $1/e$. This is shown as follows: the form to be fit is once again shown Equation 2.7. r_c is defined as the value for r at which $f(r) = 1/e$, expressed in Equation 2.8. It follows in Equation 2.9 that this value r_c is then equivalent to the correlation scale λ_c. This trick implicitly assumes an exponential structure of the autocorrelation function, and is often utilized due to its simplicity in computation [19] [20] [21] [18].

$$f(r) = e^{-\frac{r}{\lambda_c}} \tag{2.7}$$

$$e^{-\frac{r_c}{\lambda_c}} = e^{-1} \tag{2.8}$$

$$\frac{r_c}{\lambda_c} = 1 \tag{2.9}$$

2.3 Machine learning in turbulence physics

The growth of the fields of Artificial Intelligence (AI), Machine Learning (ML), Data Mining, and Big Data in recent decades has propelled many areas of research in astrophysics and neighboring disciplines [22][23]. These terms are often colloquially used interchangeably, but have notable distinctions. AI is often thought of as the superset of the other terms, referring to software or other systems that exhibit behavior akin to natural human intelligence. Machine learning systems are a subset of this, using self-learning algorithms to improve performance through experience, in the form of large amounts of data. Through the use of machine learning, a programmer or user does not need to specify much previous knowledge outside of network hyperparameters or architecture, thereby allowing the network to build its own representation of the task [4].

The idea of deep machine learning, referring to the depth and layers of abstraction in network architectures, dates back to the 1940's and its popularity in research and applications has grown and shrunk over the years. With recent leaps in modern computing hardware and data storage capabilities however, deep learning is now at the forefront of many research fields [4].It is through this increase in parallel processing capabilities and improved storage capacities that machine learning is often associated with the fields of data mining and big data. Data mining is regarded as the process of extracting meaningful patterns from raw data through relevant data collection and processing methods; the method of extracting these patterns often involves ML algorithms. Big data is regarded as an umbrella term for data mining methods and processes, dealing specifically with methods of efficiently refining large databases of raw data. In physics, the focus of data mining is often Knowledge Discovery in Databases (KDD) [22]. The exponential increase in new data being measured was especially felt in astronomy and space physics and has brought forth the idea of a fourth paradigm of research through often automated computation with large sources of data, with the first three paradigms encapsulating theory, observation, and simulation [22]. With such complex and multi-scaled interactions within turbulence studies, this new paradigm is now widely utilized.

In space physics, and solar wind turbulence in specific, current research is largely still focused on uncovering correlations, patterns, and interactions of physical properties and characteristics. Like many big data applications, this form of investigation includes data processing and reduction, but there also exists a need to translate findings into digestible physical knowledge and understanding that can either be validated through existing knowledge from the first three research paradigms or used to create new knowledge [22]; this is not unlike data mining in marketing and presenting findings to shareholders. Turbulence physics is filled with statistical laws, spectral power laws etc., can be verified and explored using modern machine learning and data mining techniques; more importantly, new relationships are sure to be discovered [22]. The relationships between structures of data in the real space and in the spectral space are often also of importance in turbulence studies, as well as the geometrical characteristics of these structures [22].

[22] outlines the general framework for data mining in turbulence research:

1. Data collection: the associated processes in collecting and creating data, or querying existing data, as well as combining data sets through "data fusion." A notable instance of data fusion is in NASA's OMNI data set [24][25], which is a multi-source data set using data from spacecraft like WIND, ACE, and IMP 8. OMNI is widely used in solar wind and plasma studies due to its long time coverage over the years and wide range of measured variables [25].
2. Preprocessing: refers to the cleaning, refining, and preparation of data so that it may meaningfully be used in the given task. Certain algorithms require specific types of encodings of variables, including numerical, categorical, and one-hot encodings. Additional considerations must be made when encoding non-binary categorical variables as incorrectly nominal/ordinal encoded features can adversely affect learning. Data transformations also fall under preprocessing, including removal of missing (NaN) values, interpolation, binning, varaible construction, Fourier transformations etc. Normalization/standardization of variables is also common for use in machine learning algorithms.
3. Attribute selection: often times, the performance of machine learning algorithms is limited by an effect named the "curse of dimensionality." This phenomenon is well known within data science and adjacent disciplines, and is a result of the feature space volume growing too fast with a greater number of features. When feature spaces are large, the available data becomes very sparse, preventing machine learning algorithms

from generalizing well to unseen data. To prevent this, a process of dimensionality reduction is often performed. Attributes/features can be carefully selected from the full set of features in a process called feature selection; many options exist for feature selection, including forward/backward selection and algorithm-wrapped selection. While reducing the set of original features is a straightforward form of dimensionality reduction, the same benefits can be gained through feature construction, using techniques such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA). Constructed features can then be used in place of original features while preserving the majority of data variance.

4. Algorithm selection: often categorized into unsupervised, supervised, and semi-supervised machine learning. Considerations include algorithm suitability to the task, complexity, and explainability. Types of algorithms frequently used in space physics will be outlined next.
5. Optimization: tuning of model hyperparameters through cross-validation performance to improve model generalization. Cross-validation, which sequentially holds out a subset of data for validation to obtain an aggregate measure of performance, helps alleviate the effect of the bias-variance trade-off in machine learning models; techniques like regularization also seek to reduce this effect through a model complexity penalty. The effect of preprocessing choices can also be investigated.

Machine learning algorithms can generally belong to one of three categories: unsupervised, supervised, and semi-supervised machine learning; all three are applicable to turbulence research.

Unsupervised methods are typically exploratory in nature, in the sense that a target variable is not imposed and the goal is more focused on exploring patterns in the given data. Unsupervised algorithms require minimal input from the designer in terms of hyperparameters, which also limits the degree to which existing knowledge can be incorporated. Examples include clustering and dimensionality reduction [23][22]. Supervised methods instead impose a "ground truth" target variable to which the a mapping is derived from the input variables, which can then be used to predict/classify unseen instances. Supervised algorithms can be more involved in the optimization process, allowing for tuning of more hyperparameters and choices of validation methods. To test model generalizability, a subset of data, called the test set, is set aside during the training phase, and is only seen after training to evaluate performance. This is standard as certain types of models are susceptible to overfitting on training data, that is to say performance is high on seen data, but significantly worse on unseen data. Semi-supervised methods strike a middle ground in which some prior knowledge can be incorporated while still using letting the data speak for itself.

While in-depth explanations of each type of algorithm application discussed is outside the scope of the literature review, it is important to review state-of-the-art uses of machine learning in astronomy, space physics and related studies.

2.3.1 Unsupervised

Amaya et al.[5] implements an unsupervised machine learning approach for classifying solar wind types in an effort to automate repetitive manual analysis. Solar wind has generally been categorized as either fast or slow, with fast winds reaching mean velocities of $750 km/s$ and slow winds reaching maximum velocities of around $500 km/s$. However, solar wind measurements can also determine its type of origin through analysis of its ion composition. This work therefore considers four types of solar wind categories: fast solar wind of coronal hole origin, slow solar wind of non-coronal hole origin, solar wind of streamer belt origin,

and solar wind of sector reversal origin. Data is sourced from hourly measurements from the Advanced Composition Explorer (ACE) spacecraft over the years 1998 to 2011.

Various empirical thresholds are used as constructed features following theory from the literature, resulting in a total of 15 features in the data set. Due to missing or low quality data, 51,374 usable data entries are available from the total 122,712. Features are then min-max normalized between 0 and 1 through Equation 2.10.

$$f = \frac{F - F_{min}}{F_{max} - F_{min}} \tag{2.10}$$

The work focuses on a self-organizing map method, and uses a k-means and Bayesian Gaussian Mixture (BGM) clustering result for comparison. Prior to clustering, two dimensionality reduction techniques are considered: Kernel Principal Component Analysis (KPCA) and autoencoders.

Principal Component Analysis (PCA) is widely used in data science to reduce feature space volume through creating a new feature space consisting of linear combinations of original features. Formally, these features are eigenvectors of the covariance matrix of the original (centered) data. The result is a set of independent, perpendicular axes; axes with the highest corresponding eigenvalues capture the most variance contained in the original data. Often times, PCA is used to reduce the feature space to a fraction of the original space with minimal information loss. However, an intrinsic limitation of PCA is that the resulting features are necessarily linear combinations of original features, which can often be too restrictive. Instead, KPCA can be used to also consider non-linear combinations. KPCA acts by transforming the data into a high-dimensional space through a kernel function, from which regular PCA can be used to linearly separate resulting clusters. The kernel function allows for the non-linearity that regular PCA cannot produce; popular kernel choices include polynomial kernels and Gaussian kernels. In this work, an eighth-degree polynomial kernel is used.

Autoencoders are machine learning models primarily used for applications related to generation or denoising, but can also be used as a dimensionality reduction technique. Autoencoders are trained to reconstruct their inputs in their outputs through an encoder-decoder process in which an underlying non-linear latent space is discovered; naturally, reconstruction error is used as a loss function. More formally, an autoencoder consists of an encoder network f and a decoder network g as shown in Figure 2.7, mapping an input vector x to its reconstruction r. The latent layer h is often of a much lower dimension compared to the input vector x; if the decoder network g is able to effectively reconstruct the input vector with minimal error, the lower dimenionality of the latent layer can be thought of as a lower dimensional representation of data points. As the encoder network is a fully connected feed-forward network, the process f is non-linear [4].

The two dimensionality reduction methods are then tested with the two clustering methods. [5] cites publications that use k-means and BGM clustering in the field of solar wind classification, but none utilize the proposed dimensionality reduction techniques. Figure 2.8 shows the results of these baseline classification techniques; the top row consists of k-means clustering results and the bottom row consists of BGM results. Each column corresponds to the data transformation techniques: the min-max normalized original data, KPCA-transformed data, and autoencoder-reduced data. Axes are chosen to provide clear visualization of the multi-dimensional data; naturally, the axes that show the most data variance for the KPCA-transformed clustering are the first two components. An important observation is that results differ with the two methods in both the KPCA and autoencoder-transformed data. Particularly with KPCA-transformed data, the clustering methods seem to struggle with the homogenous data and find Voronoï regions as clusters. This is a shortcoming of classical

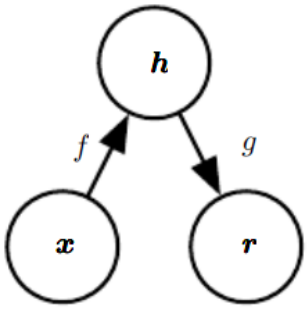


Figure 2.7: Schematic structure of an autoencoder: an input vector x is reconstructed as output vector r through a bottleneck latent layer h. The process f is an encoding network to the latent representation h, and the process g is a decoding layer used to deconstruct the code h. If trained effectively, the encoding network f can be used as a method of dimensionality reduction [4]

clustering techniques as data densities are not fully considered. As a result, clustering results can be implementation and seed-dependent. This work mitigates this effect by running the k-means and BGM algorithms 100 and 30 times respectively to arrive at a global optimum solution. The use of DBSCAN, a density-based clustering technique, is proposed, but not recommended due to the homogeneity of data. Often, unsupervised clustering methods can be evaluated on classification accuracy, however a ground truth label for each data point is not available to do so for this work.

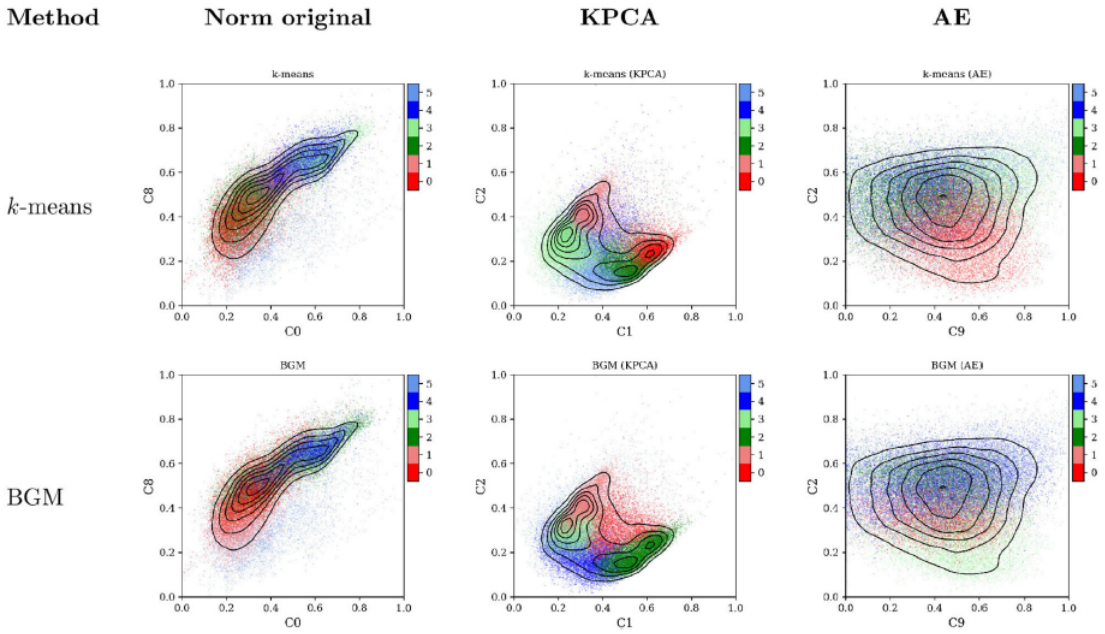


Figure 2.8: Comparison of k-means and BGM clustering techniques with different data transformations. Colors are randomly assigned to clusters, and black lines represent isocontours of data density [5]

The main contribution of the work is the use of a Self-Organizing Map (SOM). SOMs are unsupervised clustering/dimensionality reduction methods, often using a two-dimensional node lattice associated with weights called "code words." The position of these nodes is

automatically determined in the learning process in which weights are iteratively shifted towards input data points based on a distance metric. The result is that code words tend to areas of high data density; a data point's label is naturally the closest associated code word. One can specify a number of clusters beforehand, but SOMs can also determine this automatically. [5] goes into more detail on the theoretical workings of SOMs and their dynamic variant.

The work concludes that the unsupervised clustering methods investigated must be paired with suitable verification procedures, including time series visualization and SOM lattice visualization. Additionally, these techniques must be complemented with suitable data processing choices, including domain-informed feature construction. Feature construction choices are dependent on the goal of classification; if one aims to classify solar origins of solar wind, features related to solar activity should be considered. The authors also advocate for a ground truth database for such classification tasks in order to better evaluate potential classification methods.

2.3.2 Supervised

[6] explores supervised classification of near-earth plasma regions through Convolutional Neural Networks (CNNs). Similar to with [5], classifications of near-earth regions are necessary for statistical studies of solar winds; however this classification is traditionally done manually through threshold-based classifications. Supervised machine learning is therefore investigated as a method to automate this process and minimize the likelihood of misclassified or unclassified data.

The work uses data from NASA's Magnetospheric Multiscale (MMS) mission , launched in 2015, which consists of a cluster of four spacecraft in a tetrahedral formation. Such a formation allows for the investigation of spatial discrepancies for multi-scale interactions in Earth's magnetosphere. Most notably, the MMS data pipeline includes a "scientist-in-the-loop" annotation system in which an expert selects data intervals of interest and labels the relevant characteristics, including the region of occurrence; this provides ground truth labels for supervised training. The ten region categories include:

- Solar wind (SW)
- Ion foreshock (FS)
- Bow shock (BS)
- Magnetosheath (MSH)
- Magnetopause (MP)
- Boundary layer (BL)
- Magnetosphere (MS)
- Plasma sheet (PS)
- Plasma sheet boundary layer (PSBL)
- Lobe (LOBE)

Any interval without an accompanying annotated label is rejected. A total of 8,437 data points were collected, using some emphasis sampling for lesser occurring labels. Data instances contain 12 features:

- Magnetic field magnitude and its three components
- Velocity megnitude and its three components
- Ion density
- Total temperature and its two components

Each data instance is in essence its own multi-dimensional time series, with the series resampled on the cadence of the lowest resolution measuring instrument, the Fast Plasma Investigation instrument, at 4.5 seconds. The task is therefore a time-series classification

problem. Time series often have local dependencies or correlations due to underlying phenomenon, in this case the plasma region. An effective and popular supervised machine learning model for such tasks is a Convolutional Neural Network (CNN). CNNs differ from traditional artificial neural networks with the inclusion of convolutional layers; these layers perform a mathematical convolution operation through a learned kernel of weights. A nonlinear activation function typically follows. Convolution layers bring important advantages to a neural network architecture, namely having sparse interactions and parameter sharing [4]. Sparse interactions refers to the advantage of having a kernel size much smaller than the input size, requiring fewer parameters to be stored but still being able to detect underlying features in spatially dependent data. Unlike dense (fully-connected) neural networks, where each neuron (weight) is 'tied' to a single input parameter, a kernel in a convolution layer is used on the entire input, disregarding edges depending on the implementation; as a result, only one set of parameters is needed for the whole input. This is referred to as parameter sharing, and also reduces the memory requirements of a network [4]. To allow convolution to be equivariant to certain input transformations, that is to say the output changes in the same way as the input, pooling layers are often used. Without a pooling layer, convolutions are not equivariant to transformations like scaling or rotation; being so allows the network to detect motifs in the data despite orientation or position along a time series. Pooling layers act on the output of a convolution layer to aggregate values using a summary statistic within the kernel size [4]; these can use the (weighted) average, maximum etc.

The work uses a CNN of the following structure:

- 3 convolution blocks consisting of:
  - Colvolution layer(sizes $128 \times 8 \times 8$, $256 \times 5 \times 5$, then $128 \times 3 \times 3$)
  - Batch normalization
  - ReLU activation function
- Global average pooling
- Softmax layer

Rectified linear unit (ReLU) activation function layers are defined as $max(x, 0)$, where x is the output of a linear transformation in a neural network [4]. Since the ReLU function is fairly close to being linear, it retains the ease in optimization in gradient descent problems, while still introducing sufficient non-linearity to the network to model nonlinear effects [4]. Batch normalization simply acts to normalize the raw data between layers within one training batch. A softmax layer is an activation function to convert an output to one useful for classification; inputs are mapped according to Equation 2.11, where the multiplications of the true value t_i and the log of the predicted value $f(s)_i$ are summed across all classes C, resulting in "probability" values between 0 and 1 for each label.

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{e_j}} \tag{2.11}$$

A softmax output is used in tandem with a categorical crossentropy loss function for training. Such a loss function not only aims to maximize the output value for the corresponding correct label, but also minimizes the values for corresponding incorrect labels. More specifically, crossentropy is defined with Equation 2.12; through substituting $f(s)_i$ with that from a softmax output in Equation 2.11, the expression for categorical crossentropy in Equation 2.13 is obtained [4]. Note that since t_i is one-hot in a single-class classification problem, only the contribution of the positive class is considered in the expression; e^{s_i} is therefore replaced by e^{s_p}, the associated term for the *positive* class where $t_p = 1$ [4].

$$CE = -\sum_i^C t_i \log(f(s)_i) \tag{2.12}$$

$$CCE = -\log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) \tag{2.13}$$

Performance is evaluated in terms of classification accuracy, precision, recall, and F1-score, given in Equation 2.14 through Equation 2.17 in terms of true positive rate (TP), true negative rate (TN), false positive rate (FP) and false negative rate (FN).

$$accuracy = \frac{TP + TN}{N_p} \tag{2.14}$$

$$precision = \frac{TP}{TP + FP} \tag{2.15}$$

$$recall = \frac{TP}{TP + FN} \tag{2.16}$$

$$F1 = 2\ \frac{precision \times recall}{precision + recall} \tag{2.17}$$

Additionally, the area under the receiver operating characteristic (AUC) is used as an evaluation metric. Figure 2.9 shows a summary of the CNN performance. The authors note that the FS, MSP, and LOBE classes exhibit high performance metrics due to the unique patterns present in the respective regions. Next, they note that BS, MP, BL, PS, and PSBL exhibit lower performance. BL and BS regions in specific are thin boundaries between larger layers; some error may be at play with expert annotations in the MMS data pipeline due to the potential overlap of thinner and larger layers within one time series interval. A confusion matrix in [6] supports such an explanation, as classification errors are most common between adjacent regions. In physical terms, this is also justified as plasma properties of similar regions are likely to be similar.

	Precision	Recall	f1-score	AUC
SW	0.88	0.92	0.90	1.00
FS	0.93	0.95	0.94	1.00
BS	0.85	0.80	0.83	0.99
MSH	0.91	0.91	0.91	0.99
MP	0.84	0.84	0.84	0.99
BL	0.84	0.84	0.84	0.98
MSP	0.96	0.95	0.96	1.00
PS	0.89	0.89	0.89	0.99
PSBL	0.86	0.87	0.87	0.99
LOBE	0.95	0.95	0.95	1.00
Macro average	0.89	0.89	0.89	0.98
Micro average	0.89	0.89	0.89	0.99
Accuracy	0.89			

Figure 2.9: Summary of CNN performance. A macro average refers to an unweighted mean. A micro average refers to a weighted mean by class label occurrence [6]

An investigation into feature importance was also performed in the form of a Class Activation Map, which highlights sections of the various input time series that contribute to the final prediction of the CNN. For some classes, such as the SW class, large portions of the time series are pertinent to the result, indicating the values of features are important.

Conversely, some classes like the FS class is more sensitive to changes in values in the form of local maxima or large gradients than the values themselves. Finally, the learned CNN model is tested on the Cluster mission's data with favorable results.

[7] presents a machine learning approach using gradient boosted regression to predict solar wind speeds near earth. Prediction and forecasting of solar wind speeds is essential in space weather research, and is an especially important factor in monitoring solar storms and the potential effects these have on Earth's infrastructure. Currently empirical models are used which relate solar activity on the Sun to resulting events near Earth; these models include the Wang-Sheely-Arge (WSA) model, which is widely used in space weather prediction and the monitoring of coronal mass ejections. However, various sources of uncertainty exist within these models. For instance, the WSA model requires the topology of the coronal magnetic field of the Sun; typically data for only the Earth-facing side of the Sun is available. To address this, the Air Force Data Assimilative Photospheric Flux Transport (ADAPT) model, which produces multiple solutions based on possible configurations of missing data. As this method provides an ensemble of solutions, a robust method of parsing these solutions is not currently available. This work attempts to couple current empirical models like ADAPT with a gradient boosting regression ensemble model to arrive at an operational useful result and address potential uncertainties.

Gradient boosting models are ensemble machine learning models which use many weak models which, in conjunction, create a potent predictor. These weak learners are often decision (regression) trees, here with depth 3. The tree structures provide more transparency into the inner workings of predictions, in comparison to neural networks, which are commonly thought of as black-box models though some interpretability techniques have been developed for such models [26]. Additionally, a feature selection procedure is used; feature selection refers to the reduction of the full feature set into a minimal set that still produces sufficient performance; advantages include decreases in training and execution time as well as reduced model complexity and increased interpretability. In this work feature selection is performed through both observing ranked feature importance metrics and iteratively removing features from the full set; the original set of features is reduced from 1,395 features to 33 without compromising performance. These features included possible solutions from ADAPT. Figure 2.10 shows a subset of predicted solar wind results plotted over selected years and the accompanying root mean squared error; more detailed result tables comparing different feature set selections are further provided in the the work [7].

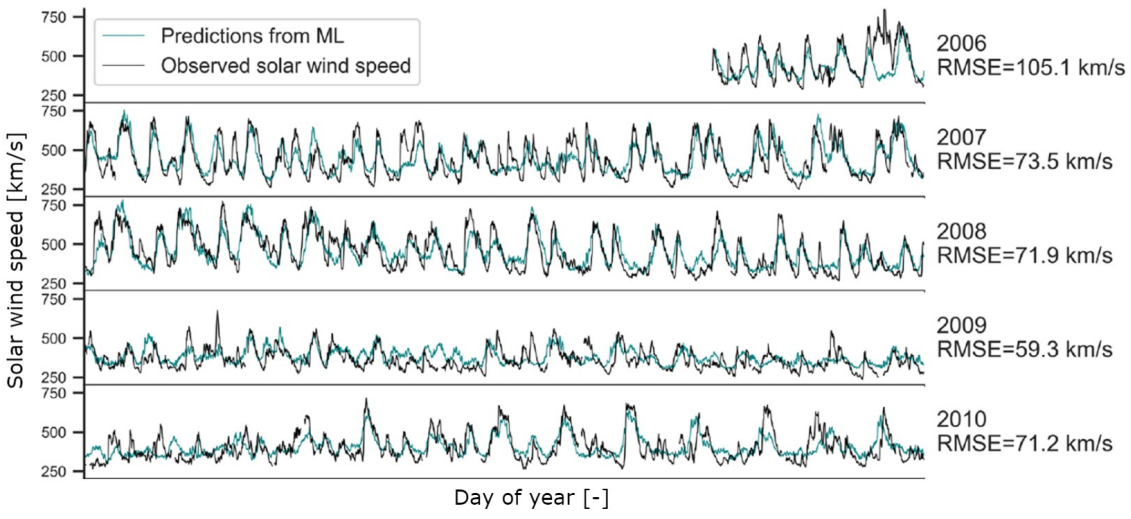


Figure 2.10: Subset of years of predicted solar wind speed compared to observed speeds [7]

2.3.3 Machine learning with spectral data

As spectral data is common in space weather studies [22], machine learning techniques that accept inputs in the spectral space are also of interest. [8], implements a one-dimensional CNN to extract soil properties from near-infrared spectroscopy data. While not in the domain of space physics, the methods in this work show promise in the possibility of automatic feature extraction that can be extended to the field. Specifically, the work addresses the prediction of oxalate-extractable phosphorus (Pox) content in soil as an aid to monitor phosphorus deficiency, which decreases crop yield. The performance of the proposed method is then compared to existing partial least squares and random forest methods.

The architecture of the one-dimensional CNN is detailed below. The input is of size 1981×1 following the 1981 spectral bands used as input. Naturally, the convolution filter sizes are also one-dimensional. Dropout layers randomly deactivate nodes in a paired fully connected layer with a specified probability for the given training batch; these help avoid overfitting the nodes in fully connected layers. The output layer is a single regression value, resulting in a single output node. A schematic the CNN architecture is also shown in Figure 2.11.

- Convolution layer: 32 filters, size 20
- Max pooling: size 2
- Convolution layer: 64 filters, size 20
- Max pooling: size 5
- Convolution layer: 128 filters, size 20
- Max pooling: size 5
- Convolution layer: 256 filters, size 20
- Max pooling: size 5
- Dropout: 0.4
- Fully connected: 100
- Dropout: 0.2
- Fully connected: 1

The data set used is fairly small at $n = 318$, with 80 entries reserved for testing and the rest used for training. Figure 2.12 shows a summary of predictive performance of the one-dimensional CNN compared to the existing partial least squares and random forest methods. The proposed method has a relative root mean squared error improvement over partial least squares of 4.4% and over random forest of 23.8%.

The trained CNN model was then used in a sensitivity analysis to observe which wavebands are pertinent to the predicted output. This analysis revealed band regions that correspond with known regions of interest from previous partial least squares models. The authors conclude that the proposed model shows great promise, but needs a larger data set to refine performance to be applicable outside of local measurements.

[9] takes a spectral processing approach to classification of stellar spectra from the Large Sky Area Multi-Object Friber Spectroscopic Telescope (LAMOST) telescope using a deep neural network. The proposed deep neural net outperforms conventional methods in classification accuracy, however an interesting byproduct is the possibility to automatically repair defective spectral data. With data from sources like LAMOST, wavelength data reconnection from individual channels can often produce defective spectra, possibly causing misclassifications. While human experts can easily identify these defects, using large volumes of data calls for an automated process to correct these defects. To solve this, a deep neural network is also used. Synthetic defective data is created through the addition of random offsets within a spectrum; the defective spectra are then used as inputs with the original, unaltered spectra are used as the output. The hope is that the neural network can then learn

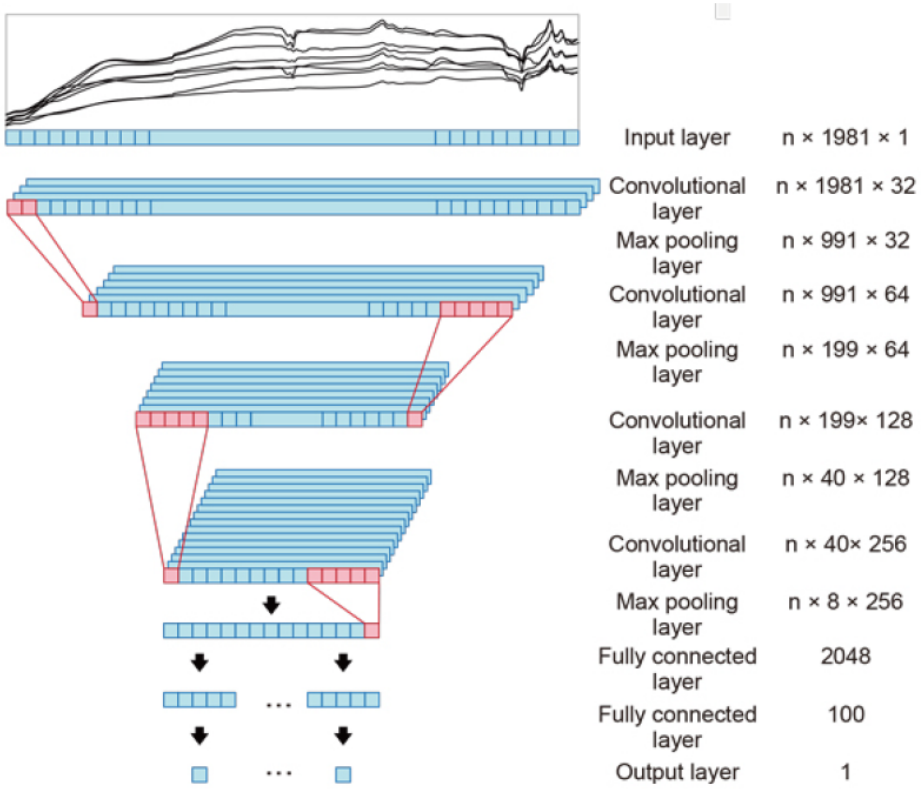


Figure 2.11: Diagram of one-dimensional CNN structure, given a 1981 long spectral band input [8]

to repair these defects. With the prevalence of missing data in space-related missions, such functionality is valuable [22] [27]. Examples of the ability to repair defective data are shown in Figure 2.13

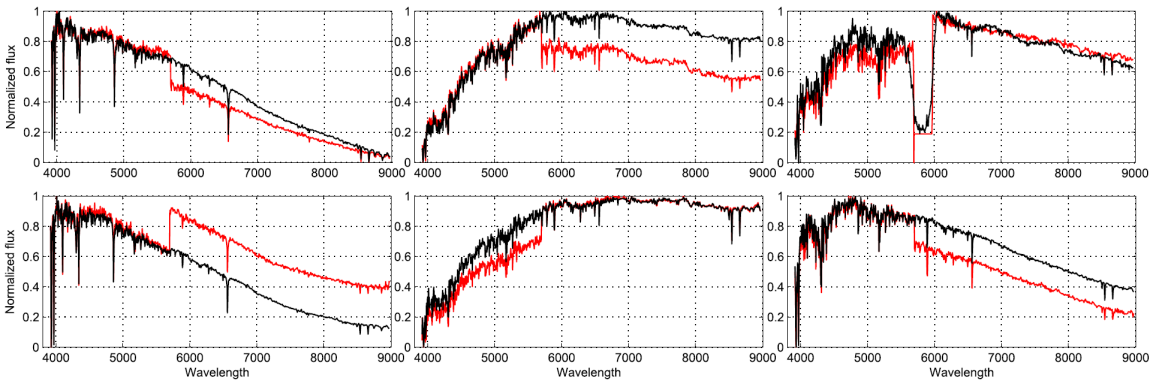


Figure 2.13: Results of automated spectra defect repair. Pseudo-defective spectra are marked in red, with the reconstruction in black [9]

[10] takes a more complex approach to classifying astronomical spectra through a multi-model voting system in which the most common prediction from a committee of three neural networks is taken as the final classification. The work takes a novel approach through using both partial and general features; here, partial features refers to local areas of interest in a given spectrum, while general features refers to the overall shape or trend of a spectrum.

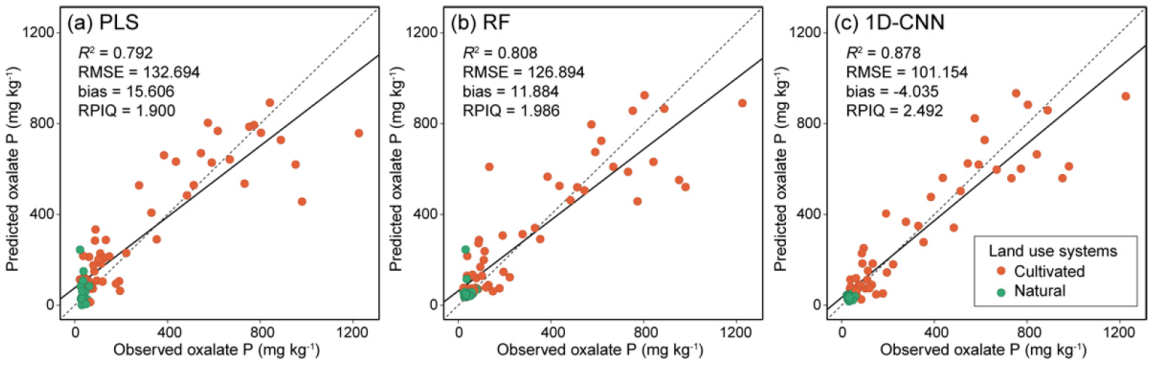


Figure 2.12: Predicted versus observed Pox content with partial least squares, random forest, and the proposed one-dimensional CNN method. Performance metrics are also provided [8]

Partial and general features are passed as input to two of the voting neural networks, with the third taking the raw data.

Partial features are obtained through first establishing a way to sparsely represent the stellar spectra through PCA, then using an inverse-transform reconstruction as a method of denoising the spectrum; this aims to preserve large spikes while smoothing noisy areas. The work discovers that the spectra can indeed be represented sparsely, i.e. represented in a reduced set of features, as shown in Figure 2.14. The raw spectra contain 2,522 features, while from Figure 2.14, it is evident that they can be represented in a far lower dimensional space. The first step in partial feature extraction is to therefore transform spectra to a feature space defined by the first 12 principal components, then inverse-transforming to retrieve denoised spectra. Next, the resulting spectra are differentiated as to emphasize local spikes; the denoising step prior to this helps reduce spikes resulting from noise. The resulting spiky spectra now contain the partial features of interest, and can be transformed to a lower dimensional representation through PCA. This process is shown in Figure 2.15.

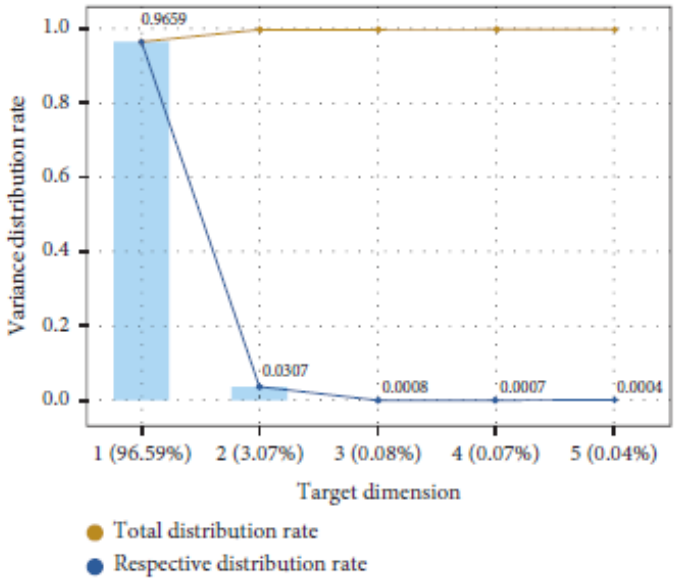


Figure 2.14: Representation of explained variance of top principal components fit on 6000 spectra. As most of the data's variance can be captured with a set of top ranking principal components, the spectra can be said to be sparsely represented [10]

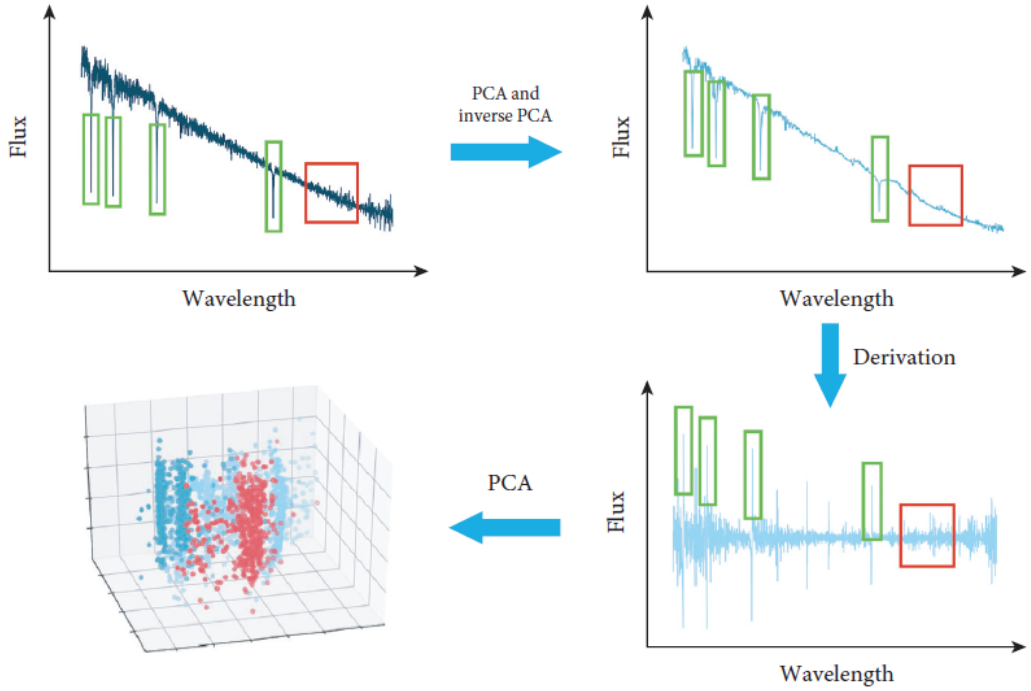


Figure 2.15: Process of extracting partial feature vector. Areas marked in green indicate partial features of the spectrum, and the area marked in red is a non-partial feature. First, the spectrum undergoes a PCA transform and inverse transform using 12 principal components. This helps denoise the non-partial features as shown in the area marked in red; partial features are preserved. The spectrum is then differentiated to emphasize large changes/spikes in the spectrum; partial features appear as peaks in this new spectrum and non-partial features stay flat. The resulting spectrum can then be PCA-transformed to reduce dimensionality and form a partial feature vector [10]

To extract the general features of a given spectrum, the authors use a sliding average pooling kernel to smooth and reduce the spectrum to a vector with 7 features, down from 3,522. As shown in Figure 2.16, the average pooling process is similar to that in a one-dimensional CNN, and describes the overall data tendency.

The final of the three voting models is a one-dimensional CNN using the raw spectral data. The CNN consists of two convolution-pooling blocks, followed by two fully connected layers including a softmax output layer. The models used for partial and general feature vectors are regular artificial neural networks with fully connected layers leading to a softmax output. The complete voting pipeline is shown in Figure 2.17.

Classification accuracies exceed 99% for all investigated stellar types with the ensemble voting system. For most cases, the voting system outperforms results from a singular network.

2.3.4 Data sparsity

A common obstacle with data mining and machine learning with space data is missing or low quality data; this can be the result of many factors including instrument failures and inconsistent coverage [22] [27]. [27] shows that spectra using data with large gaps inherit features of these gaps, and investigates solutions to performing spectral analysis when as much as 50% of data is missing. The methods investigate involve:

- Windowed averaged Fourier transforms

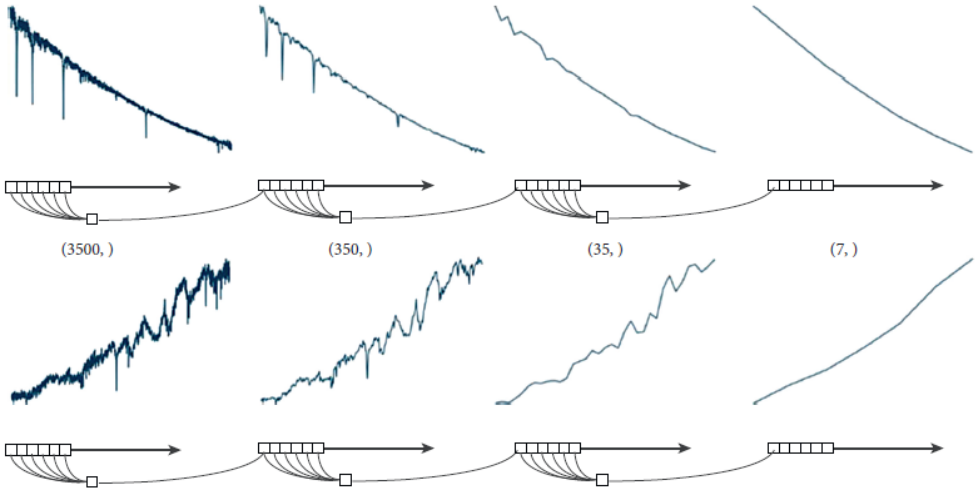


Figure 2.16: Average pooling through a sliding kernel to extract general spectrum features, i.e. overall trend [10]

- Fourier transform of correlation function (with and without linear interpolation of data)
- Maximum likelihood recovery
- Compressed sensing

Some applications find success with simple interpolation solutions such as linear interpolation of short data gaps [28] and is sometimes even favorable for neural network applications [29], while larger gaps can be reconstructed through training a machine learning model like a random forest [30].

2.3.5 Generative applications

Another subset of AI, generative networks, also sees use in the space physics domain. Generative networks, like Generative Adversarial Netowrks (GANs), can be considered forms of unsupervised or self-supervised learning, though are often used for purposes quite distinct from other unsupervised methods. GANs are trained in an adversarial manner, where a generator network and a discriminator network compete in an optimization process [4]. A generator network's purpose is to produce samples through learning a latent space probability distribution in which example data lie. A discriminator network acts to distinguish real samples from those created by the generator. Ideally, the optimization of both networks drives the generator to create more and more realistic samples through the discriminator improving its ability to distinguish fake data. An equilibrium is then reached in the form of $g^* = \arg\min_g \max_d v(g,d)$, where $v(g,d)$ is an objective function in which the generator's error is minimized and the accuracy of the discriminator's accuracy is maximized. In theory, this equilibrium occurs when the discriminator's accuracy converges to $1/2$, i.e. the generated samples are indistinguishable from real ones. The resulting generator then has great value for generative purposes, often used for image, text, or synthetic data generation [4]. In practice, this optimization process is difficult and equilibrium's are difficult to reach and unstable, and variations on the traditional GAN have been developed to address these difficulties [4].

[11] utilizes a GAN to increase spatial resolution of solar images for use in solar activity research. Current solar imaging sources suffer from either low spatial or temporal resolution, or limited uptime. Namely, images from the Helioseismic and Magnetic Imager (HMI),

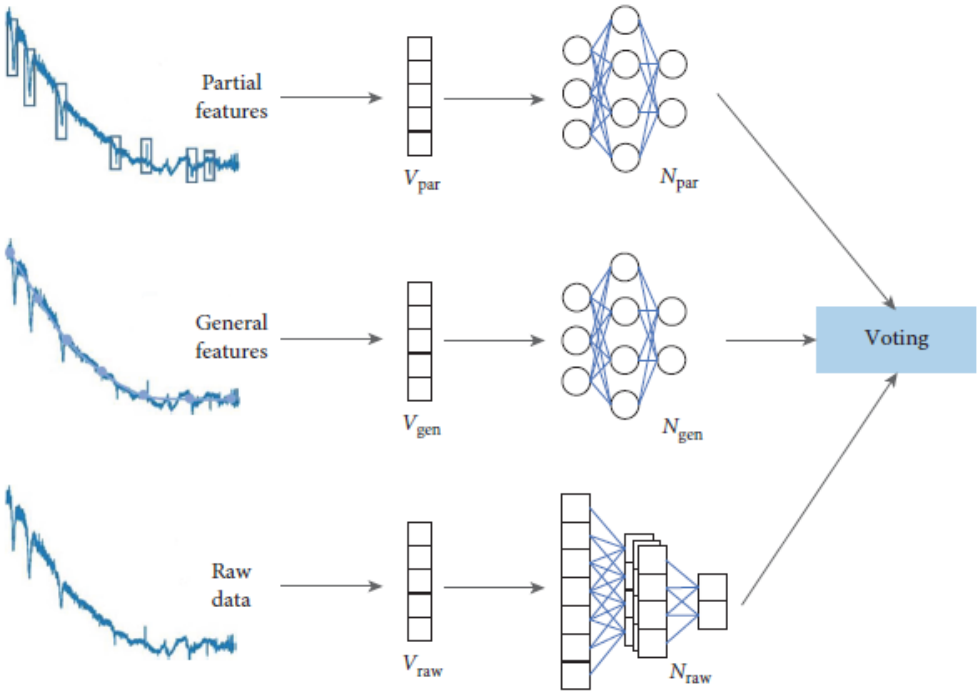


Figure 2.17: Model voting pipeline: three separate models take as input a partial feature vector, general feature vector, and raw data vector. The classification from each model is used to obtain an overall consensus through a voting system where the most common prediction wins [10]

launched in 2010, has high uptime and measurement cadence, but suffers from low spatial resolution due to instrumentation limitations. In contrast, the Goode Solar Telescope (GST) produces high resolution images with more limited availability. This apparent trade-off provides an avenue for generative networks to upscale low resolution images to make available data with both high temporal and spatial resolution. Specifically, A training set can be created by aligning and pairing observations from both the HMI and GST, resulting in pairs of low and high resolution images of the same areas. A GAN setup is then used to learn a method of increasing the resolution of low-resolution images; the setup used is shown in Figure 2.18: the discriminator is designed with the purpose of feature extraction and classification through convolution layers, while the generator network uses both convolution and deconvolution layers to extract features, then upsample the image.

From visual inspection, the results are quite remarkable, as shown in Figure 2.19, where low-resolution HMI images and their upscaled versions are compared to the high-resolution GST image. Reconstructed images are far sharper and retain edges of features like sunspots in the last two rows in Figure 2.19. The proposed system is able to upscale image resolution by a factor of 4, allowing for researchers to more effectively study flow fields and similar characteristics to monitor solar activity and its potential effects on infrastructure on Earth.

An application with similar motivations is explored in [31], in which image-to-image translation is performed with HMI data. Here, an opportunity is presented to generate ultraviolet and extreme ultraviolet images which only began being collected in the 1990's, where images from the HMI have been collected since the 1970's. This manifests as an image translation problem, where the magnetogram images from the HMI are converted to ultraviolet images, which require vastly different instrumentation to collect. To do so, a popular established GAN Pix2Pix and Pix2PixHD [32] are modified and improved for the desired task. In

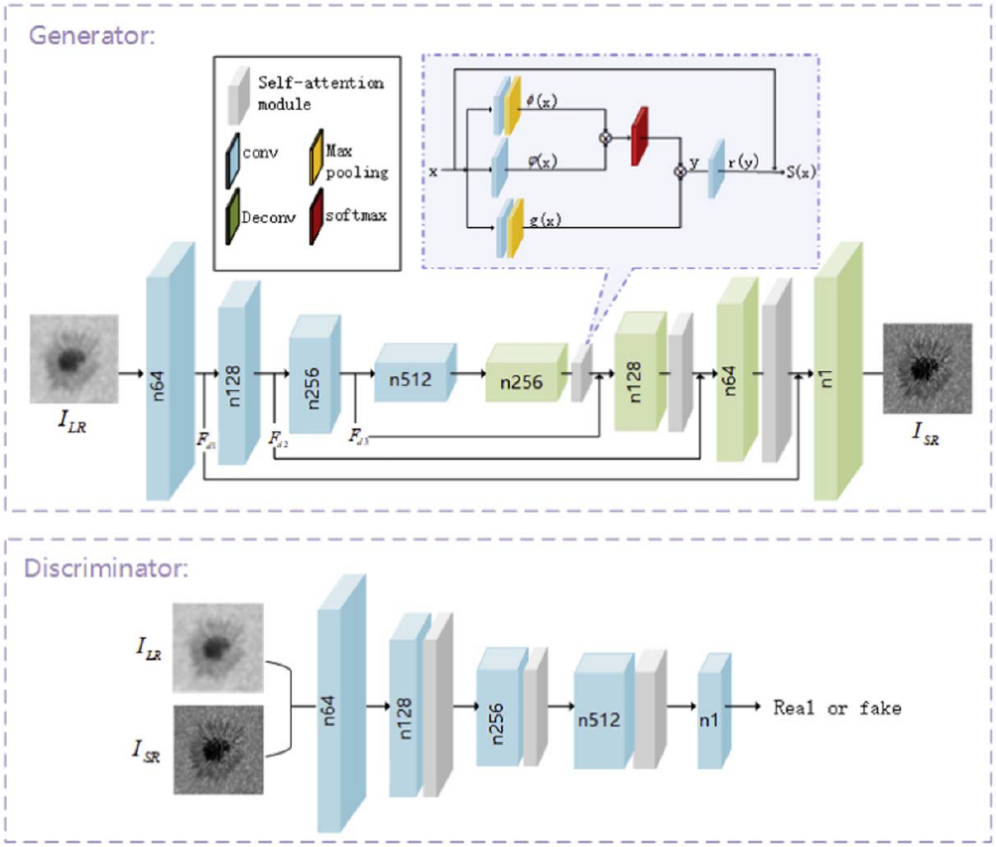


Figure 2.18: GAN structure divided in a generator and discriminator network. The generator acts to upsample the spacial resolution of solar images, while the discriminator is trained to distinguish real images from generated images [11]

a similar manner as in [31], overlapping data pairs are used to train a translation network, to be used on images where a paired ultraviolet image is not available. Examples of image translation are shown in Figure 2.20, where the translated image is visually compared to the ground truth ultraviolet image from AIA0304 images from the Solar Dynamics Observatory. Active regions visible in the HMI images correctly appear in the translate image, and high fidelity patterns are inferred from inactive regions.

The modified Pix2PixHD architecture achieves a pixel-to-pixel correlation of 0.99 and a structural similarity index of 0.96, with 82% of pixels having less than 10% error, all indicating a high performing image translation. Additionally, the modified Pix2PixHD algorithm outperforms the original Pix2PixHD and Pix2Pix systems. (Extreme) ultraviolet image data are used by researchers to study solar events and areas of interest such as coronal loops and active regions; these events have a large effect on solar weather experienced on earth through events like solar flares and coronal mass ejections, and awareness of solar activity allows for prediction of these effects. A system to reliably and accurately generate matching ultraviolet images from decades of magnetogram data is therefore extremely valuable.

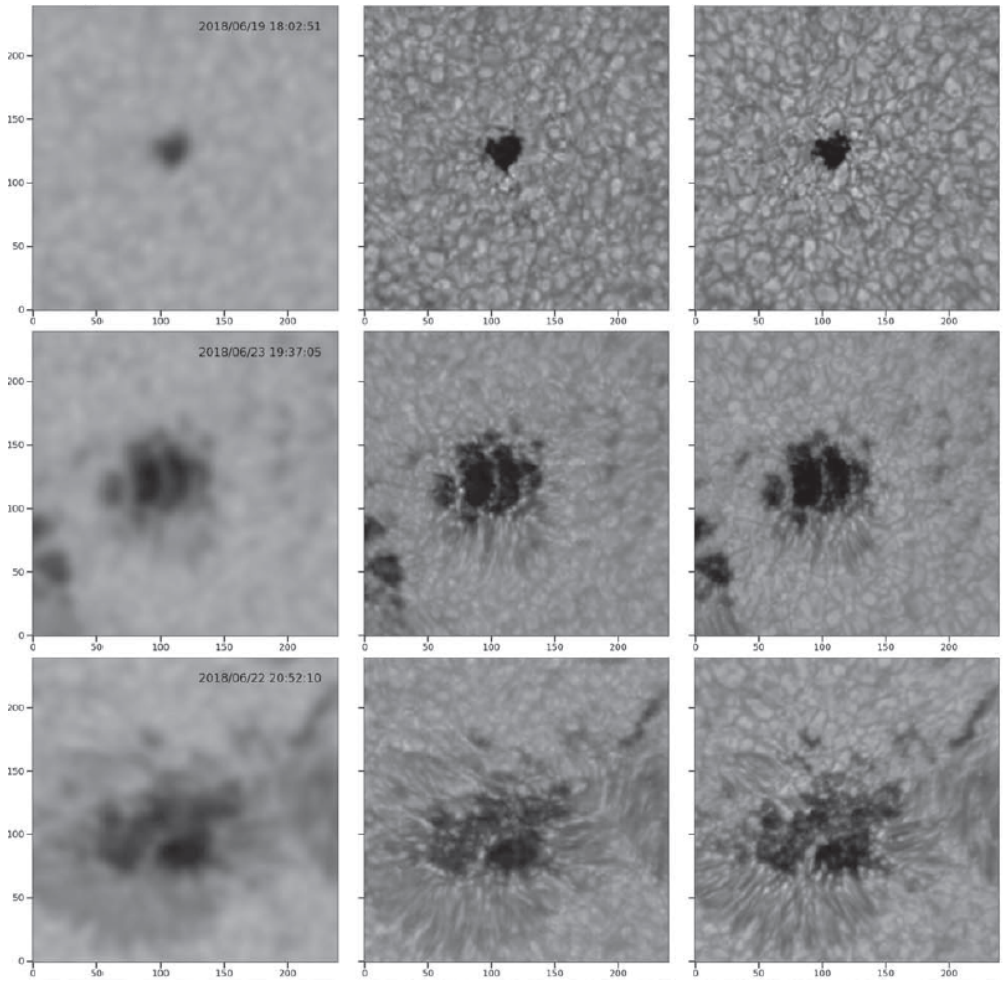
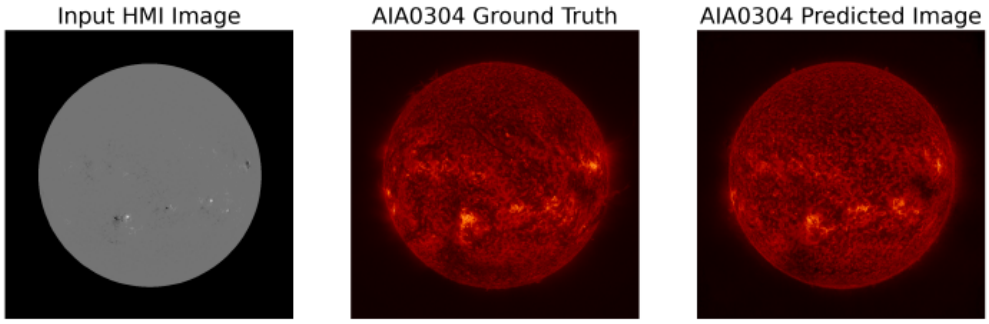
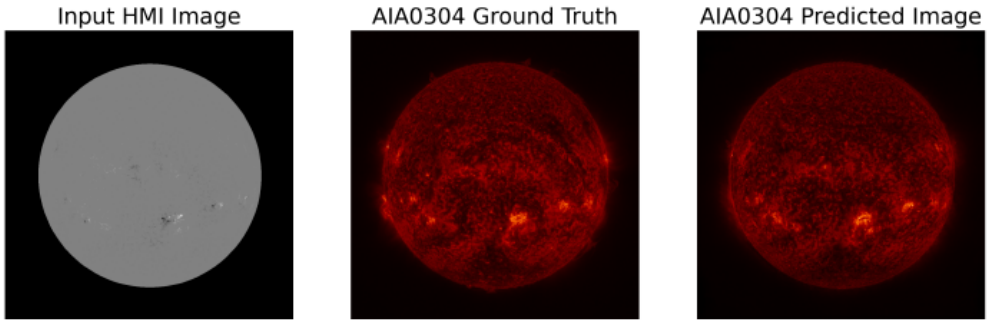


Figure 2.19: Examples of testing results: each row contains the low-resolution HMI solar image, followed by the upscaled image, then the high-resolution GST image [11]



(a) **September 01, 2014**



(b) **September 05, 2014**

Figure 2.20: Examples of test results: each row contains the HMI magnetogram image, followed by the AIA0304 ultraviolet image and the predicted ultraviolet image using the generative network [11]

Chapter 3

Background

First, some necessary background is addressed before detailing the implementation of this work in order to provide context and define terms. First, some terms and concepts that pertain to the collected solar wind data are defined, then the types of machine learning models used are discussed.

3.1 Plamsa physics

3.1.1 L1 orbit

Much of the data used in this work is collected from spacecraft in an L1 orbit of the Earth-Sun system. L1 refers to a Lagrange point, which are shown in Figure 3.1; such points exist at points where the gravitational pull of both relevant masses are canceled out by the centripetal force experienced by a spacecraft. Spacecraft in these Lagrange points therefore require reduced amounts of fuel to maintain their orbit [12]. For two-body systems, five Lagrange points exist, though the L1 point offers a constant line of sight on both masses and is the one closest to Earth, reducing communication times.

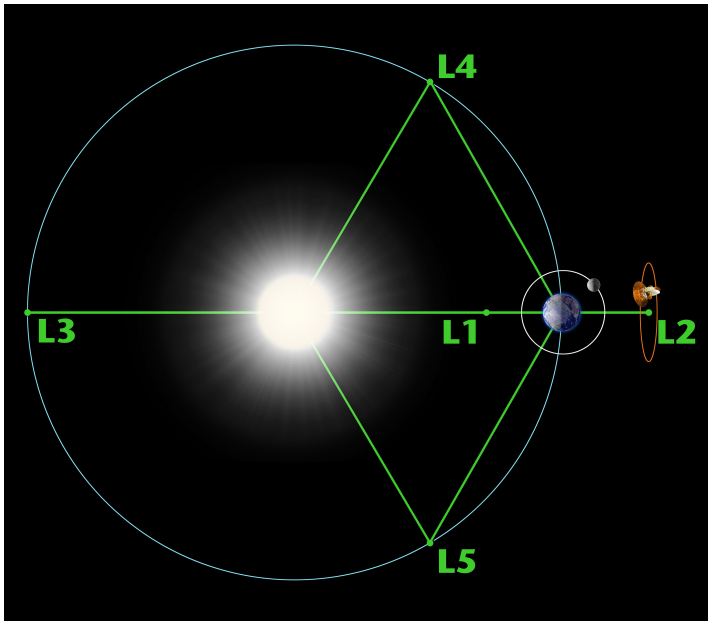


Figure 3.1: The five Lagrange points of the Earth-Sun system. The data used in this work is collected from spacecraft in an L1 orbit [12]

3.1.2 GSE coordinate system

Data collected for investigation of solar wind typically use the Geocentric Solar Ecliptic (GSE) coordinate system. This system is defined as the following: X-axis pointing from the Earth to the Sun, the Z-axis along the ecliptic north pole, and the Y axis is defined along the ecliptic, pointing towards dusk to form a right-handed coordinate system [33]. The velocity vectors used in this work are expressed in this GSE coordinate system; since solar wind originates from the Sun, often times the first component (X-component) will be the dominant term.

3.1.3 Earth bow shock

Earth's bow shock is a shock boundary that results from incoming solar wind from the Sun colliding with and being deflected by Earth's magnetosphere. The location of the nose of this shock is used by in the OMNI data set as a standard measurement location to better investigate effects between solar wind and the magnetosphere [34]; measurements are taken at L1, then time shifted to the nose of Earth's bow shock. The precise mechanisms behind determining the bow shock location and the time shifting method are further described on the OMNI data set documentation [34] and further treated in chapter 4.

3.1.4 Plasma beta

The plasma beta (β) is one of the parameters considered in this work, which is defined as the ratio between the gas/plasma pressure to the magnetic pressure [35] and is often used in studies regarding Earth's magnetic field and its interaction with solar wind.

3.2 Machine learning

3.2.1 Learning algorithms

Machine learning algorithms, together with the concepts treated in chapter 2, can be generally defined as an algorithm/model that "learns" patterns and relationships from provided data [4]. Goodfellow (2016) [4] and Mitchell (1997) [36] define such an algorithm in various aspects, which can be defined in the scope of this work.

The task A machine learning task is defined to give context to other decisions made in the machine learning pipeline. Tasks can be categorized in various ways, such as being supervised or unsupervised, classification or regression, exploratory or formative. This work's machine learning task is one of exploratory regression, with an adjacent task of discovering relationships between parameters towards a regression prediction. A regression task is characterized by the predicted feature being a numerical one. The task is exploratory in nature as the goal is to create new knowledge in a field of study relatively unexplored by machine learning techniques, rather than create an artifact to be used for its predictive capabilities. With this task in mind, a research frame is established from which to make subsequent algorithm design choices.

Measure of performance Naturally, the choice of a quantitative performance metrics has a great effect on the result of training a machine learning model, but also on the interpretation of results. A choice must be made that both aligns with the machine learning task and the behavior of the model. As an example, regression tasks optimized on a mean squared

error loss function will be more sensitive to large errors, while mean absolute error can be considered more easily interpretable. For a true measure of performance, models should be evaluated on unseen test data. As an extension, regularization methods can be used in a loss function to achieve control over model complexity and behavior, and cross-validation can be used to achieve mean performance measure.

Experience Broadly speaking, a machine learning algorithm is said to gain experience through exposure to training data. Given the nature of the treated task, this experience comes in the form of a synthesized data set made from multiple data sources, and a numerical target variable that should be predicted.

3.2.2 (Deep) multi-layered perceptron

Multi-layered perceptrons (MLPs), also called artificial neural networks due to the parallels to representations of neurons in the human brain, are among the most popular machine learning models and are used in various commercial applications [4]. MLPs are made up of perceptron layers chained together to arrive at an output value through a feedforward computation. A simple representation of such a network is shown in Figure 3.2 [4]: an input vector $\mathbf{x} = [x_1, x_2]^T$ is fed through a single hidden layer h with weights h_1 and h_2, to arrive at output y. The representation on the right of Figure 3.2 is a more compact representation, where the output of a hidden layer can be expressed as a mapping of the input to the layer through a weight vector w. Outputs of a hidden layer are then passed through a choice of non-linear activation function. These hidden layers can be of different widths (number of neurons), and then chained together to achieve a "deep" network. To achieve a given task with an MLP, the depth and width of the hidden layers must be sufficiently large to model the underlying mapping from the input x to the output y, while not being too large that overfitting starts to occur. Neuron weights are updated through error backpropagation during training with respect to a specified loss function. Of note is that the shape of the output layer y is simply a single node for the given single-value regression task.

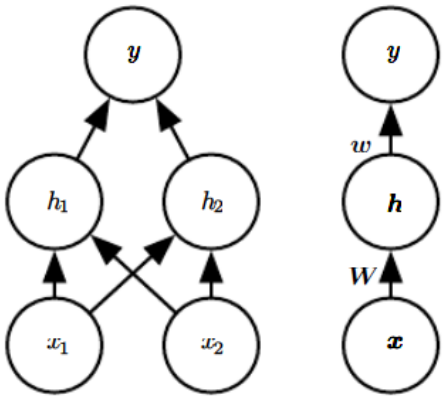


Figure 3.2: A representation of a simple feedforward MLP from Goodfellow [4]. The left representation shows the individual nodes or "neurons" and their inputs. The right representation is a more compact version using instead vector operations to represent layers.

3.2.3 Lasso regression

Basic linear regression is a simple regression model that computes a predicted output $\hat{y}$ as defined in Equation 3.1, where the input vector $\mathbf{x}$ is simply multiplied feature-wise with the

weight vector $\mathbf{w}$; an intercept b is then added. An expanded representation of this is shown in Equation 3.2. The individual weights w_i can provide insight to the contribution of the respective feature x_i to the final prediction: the sign of w_i represents the direction of the contribution and the magnitude of w_i represents the strength of the contribution.

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b \tag{3.1}$$

$$\hat{y} = w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n + b \tag{3.2}$$

Linear regression models can be made more robust to overfitting through utilizing regularization techniques that modify the loss function. A common form of regularization is the L1 norm weight decay, which adds a term to the loss function that penalizes the magnitude of network weights. The use of the L1 term specifically in linear regression is referred to as LASSO regression(Lease Absolute Shrinkage and Selection Operator) [4]. This term, $\Omega(\theta)$, is defined in Equation 3.3 [4], where the L1 norm, $||\cdot||_1$, is the sum of absolute values of network weights. An L1 norm differs from an L2 norm, which penalizes the sum of squared network weights, in that an L1 norm allows weights to decay all the way to zero [4]. A loss function utilizing an L1 norm is represented as in Equation 3.4 [4], where the original loss function $J(\theta)$ has an additional L1 term addition with a positive decay strength hyperparameter α. Depending on the strength of the decay, the regression coefficients of less important features will tend to zero, providing some insight into feature correlation and importance to the target variable.

$$\Omega(\theta) = ||\mathbf{w}||_1 = \sum_i |w_i| \tag{3.3}$$

$$J^*(\theta) = J(\theta) + \alpha||\mathbf{w}||_1 \tag{3.4}$$

3.2.4 Random forest

Random forests are form of ensemble model of decision trees, capable of efficiently handling large data sets based on a "divide and conquer" approach [37]. Decision trees operate through making sequential splits of data on decision boundaries, informed by the greatest decrease in entropy. A tree-like structure results from successive splits, with decisions at the leaves; these structures have the advantage of being inherently interpretable. Ensemble methods consist of a large number of weak regressors and output an aggregated result; generally, ensemble methods can outperform single, larger models of the same type due in both accuracy and computation time due to its parallelizable training. Much of this is due to bagging (Boostrap AGGregation), in which subsets of training data are sampled to build a set of weak decision trees as part of the ensemble; a prediction is made through an average, in the case of a regression task, of partial results from each weak predictor. This not only ensures that trees can be created in parallel, but that predictions are more stable through the aggregation of partial results [37]. Moreover, training requires few choices of hyperparameters, and studies have been done on the effect of such parameters, such as of the number of trees present in the ensemble and a point of diminishing returns from increasing this number [38].

Chapter 4

Data processing

This section details all steps taken to arrive at the data set created to achieve the research goals, including a description of data sources, collection methods, feature construction processes, and data combination processes.

4.1 Data sources

4.1.1 Wind

NASA's *Wind* spacecraft was launched in 1994 as part of a program to solar activity and it's effects on earth, and arrived at its L1 orbit in 2004. *Wind* is equipped with a wide array of scientific instruments separated into two suites: one focused on particle measurements and on on field measurements [39]. Of particular interest for this work is the Magnetic Field Investigation instrument, abbreviated as MFI, and the Three-Dimensional Plasma and Energetic Particle Investigation instrument, abbreviated as 3DP. More recently, higher temporal resolution data has become available for MFI's magnetic field vector measurements, sampled at 11Hz; at this resolution, it becomes possible to perform analyses of both large and small scale phenomena [40]. [39] gives an extensive overview of a wide variety of research areas which *Wind* data has helped advance, including plasma turbulence.

4.1.2 OMNI

Another popular and influential source of data is the OMNI database, containing resolutions of 1 or 5 minute measurement cadences. The database is a compilation of measurements taken from multiple spacecraft at Earth's L1 orbit, namely the *Ace*, *Wind*, and IMP-8 spacecraft, into a singular comprehensive source of data to be used in space weather research. Important to note is that measurements are "time shifted" to Earth's bow shock, meaning measurement time stamps are shifted by an amount such that measurements appear as they would when they reach the bow shock. Naturally, doing so ignores any potential changes that solar wind would undergo within the time shift amount; studies have been performed on the uncertainty resulting from this [41] [42] [43], but these are considered acceptable for the proposed work given the utility of such a consolidated data source. The mechanics of such shifts are further explained and justified in [34].

4.1.3 CDAWeb

Both data from *Wind* and OMNI are publicly available from the Space Physics Data Facility's (SPDF) Coordinated Data Analysis Web (CDAWeb) [44]. CDAWeb contains data from

various spacecraft and ground-investigation missions, spanning over 600 variables in a time series format [44]. Time series data can be plotted on the CDAWeb system, or downloaded in csv or CDF (Common Data Format) through an interactive interface or through FTP (File Transfer Protocol). The CDF format and associated software was developed by NASA, with the purpose of efficiently storing multidimensional data in a platform-agnostic format; interfaces exist for languages such as FORTRAN and Java, with user-made APIs for languages like Python, such as `cdflib` (`https://pypi.org/project/cdflib/`). For this work, cdf files are collected through CDAWeb's FTP interface, then processed in Python using `cdflib`.

4.2 Data set creation

Often times in data mining, the creation and processing of data becomes the most laborious stage of the overall process. In order to properly leverage the vast amounts of data available on CDAWeb, decisions must be made regarding (re)sampling resolutions, missing values etc., especially when merging multiple data sources. The following section will describe the process taken in order to process and consolidate multiple data streams from CDAWeb into a single data set to be used for data exploration and machine learning.

The purpose of such a data set is to investigate the relationships and predictive capabilities of large-scale turbulence characteristics on a small-scale characteristic, the Taylor microscale. Here, "large-scale" refers to either a large-scale flow characteristic, namely the correlation scale, or an averaged property of the flow, such as pressure or speed; averages achieve large-scale descriptions as they represent an sweeping/blanket representation of the flow in a given time interval. As such, individual data entries consist of properties measured across a given time interval; this interval is chosen to be 6 hours as this is a sufficient time period to observe large scale interactions such as the correlation scale [3] [16]. Time series are therefore split into 6-hour intervals, of which averaged properties are calculated. Correlation time scales and Taylor time scales are also calculated using the same interval. While low resolution data is sufficient for calculating large-scale parameters, including the correlation scale, much higher resolution is needed to calculate the Taylor scale; the high resolution Wind data is sufficient for such purposes [40] [39]. The data set therefore consists of three parts to be merged:

- Averaged properties from a merged OMNI-Wind data set
- Calculated correlation scales from Wind data
- Calculated Taylor scales using high resolution Wind data

The remainder of this section describes how the data set is created with 5 years worth of data. Using CDAWeb's FTP interface, the following data was collected for the years 2016 to 2020:

- `OMNI_HRO2_5MIN` (OMNI database)
  - Flow speed [km/s]
  - Flow pressure [nPa]
  - Plasma beta [-]
  - Temperature [K]
  - Electric field [mV/m]
  - Proton density [n/cc]
- `WI_H2_MFI` (Wind spacecraft), high resolution
  - Magnetic field vector [nT], in three components GSE coordinates
  - Magnetic field magnitude [nT]
- `WI_ELM2_3DP` (Wind spacecraft)
  - Electron density [n/cc]

  - Electron temperature [K]
  - Electron velocity [km/s], in three components GSE coordinates
- `WI_PLSP_3DP` (Wind spacecraft)
  - Proton density [n/cc]
  - Proton temperature [K]
  - Proton velocity [km/s], in three components GSE coordinates

These parameters specifically are chosen as they provide general descriptions of solar wind, including electron and proton properties that characterize kinetic plasmas [14]. Note that some redundancy is considered, such as collecting the proton temperature and temperature from both OMNI and Wind; this is simply done for exploratory purposes and redundancy will be removed when applying machine learning techniques. Most parameters require no further calculations to be added to the data set, with the exception of the correlation and Taylor scales, and the RMS (root-mean-squared) magnetic field. The RMS is calculated over the 6-hour intervals with the `WI_H2_MFI` magnetic field magnitude data, which is equivalent and less computationally intensive than doing so component-wise.

With the exception of the high resolution Wind data, files are available for each month of the desired time interval; high-resolution data is provided per day. `cdflib` provides a high level API to read these data sets into Python to then be converted into a `pandas` dataframe. One caveat before the `cdflib.cdfread.CDF` object can be converted into a dataframe format is that multi-values features must be manually split into their components; these are often vectors such as magnetic field strengths or velocities. Epoch strings are also converted into `pd.core.indexes.datetimes.DatetimeIndex` objects. Missing (NaN) values for the OMNI data sets are specified in [34]; some examples being 999.99, 9999.99 99999 etc. As gaps created by these missing values are small, they are simply linearly interpolated following [28] and [29].

Figure 4.1 shows an inspection of one day's worth of time series data for various example parameters from the OMNI database. The data in the top row of the figure behave as expected, however some time series exhibit behavior as shown in the remaining two rows: large spikes in data appear, with spikes being at seemingly discrete values. These values do not appear in the list of specified NaN values in [34], and are anomalous measurements. These discrete spikes are more clearly seen in a histogram format of the same day of data, shown in Figure 4.2.

From Figure 4.2, it is evident that data spikes are anomalous, as the majority of data lie in a "realistic" range informed by physics and through inspection of data. As a measure to constrain values to legitimate data points, a range of values in which data must lie is specified for each variable that exhibits such behavior; values outside this range are considered missing and are either subjected to a linear interpolation fill or, in the case that missing values make up above a certain threshold of data (chosen to be 40% following insights from studies on the effects of missing values in data analysis [30] [28] [29]), the data interval is discarded for purposes of data reliability. To illustrate this process, consider an example shown in Figure 4.3 to Figure 4.5. Figure 4.3 shows an example of an unprocessed time series of the x-component of the magnetic field vector over a one-day interval. Again, data spikes are seen at intervals of 1000. Through some physics-informed intuition, values should realistically lie around an absolute value of 0 to 10 nT, overestimated to account for potential large values. Figure 4.4 shows a zoomed in version of the time series in Figure 4.3; an underlying time series is visible in this zoomed-in range, with data spikes occurring randomly throughout. Figure 4.5 shows the result after removing values outside of a specified value range. As such values vastly outnumber occurrences of officially documented fill values, the missing value density is considered using these values; again, if over 40% of values within a given time interval are missing, the interval is discarded, otherwise the values are simply interpolated.

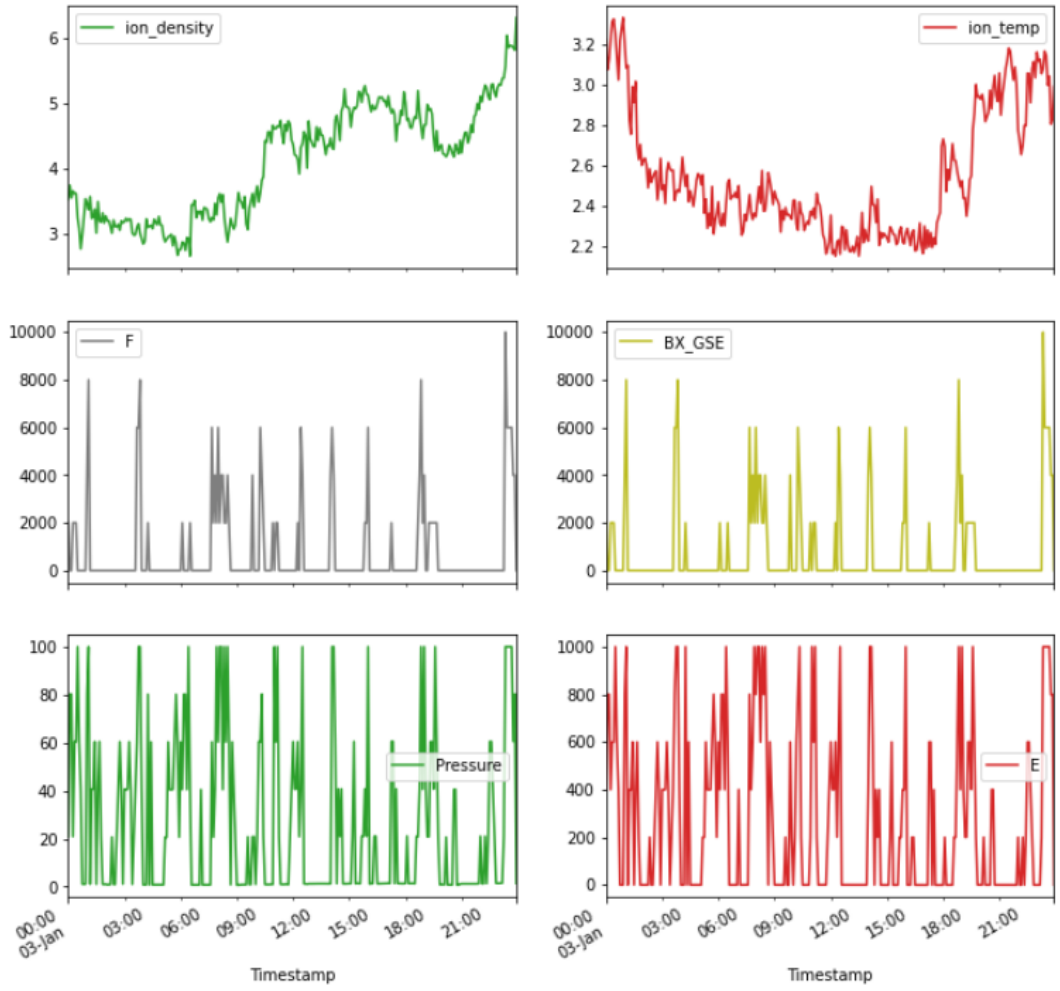


Figure 4.1: Inspection of time series data from OMNI database: ion density, ion temperature, magnetic field magnitude, magnetic field vector X component, pressure, and electric field strength

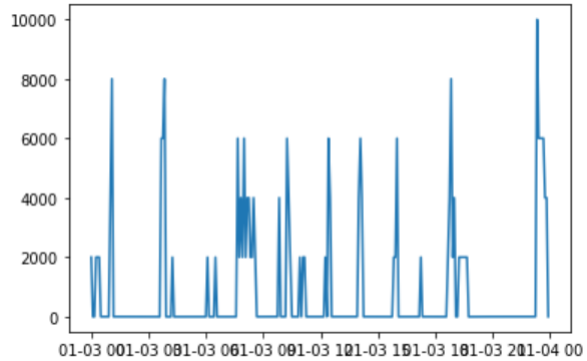


Figure 4.3: Example of unprocessed time series with anomalous data spikes: one-day interval of OMNI magnetic field vector x-component

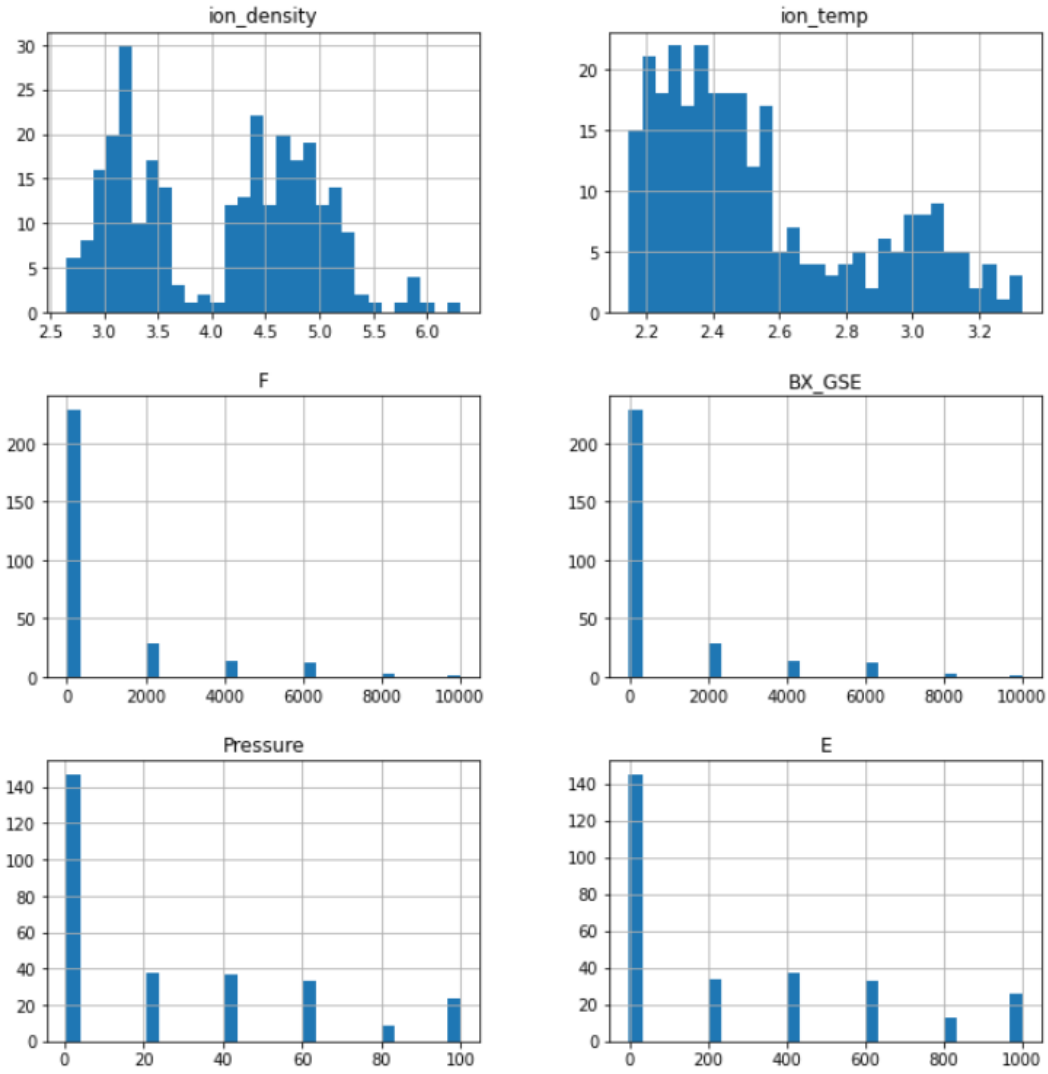


Figure 4.2: Data sample from Figure 4.1 shown in histograms. Discrete spikes are more clearly visible in the associated variables (magnetic field magnitude, magnetic field vector X component, pressure, and electric field strength), often at regular intervals depending on the order of magnitude of data.

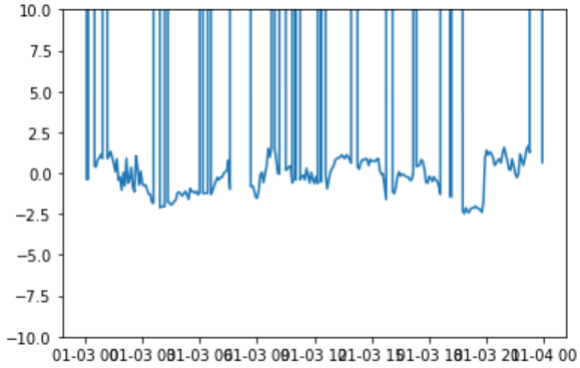


Figure 4.4: Zoomed in view of Figure 4.3, showing underlying time series data obstructed by data spikes. Values outside of a specified range, defined on a per-variable basis through inspection of data, are marked as missing

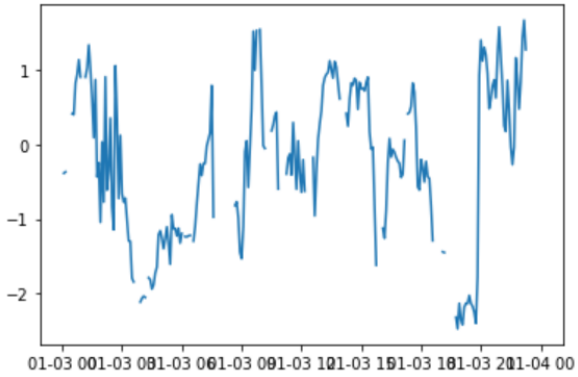


Figure 4.5: Underlying time series with anomalous values removed. If the density of data gaps exceeds 40%, the data interval is discarded for analysis

Next, the collected and corrected time series are resampled on a regular interval such that different sources may be matched up and merged into a single data set. Separately, magnetic field data is resampled to calculate both correlation scales and Taylor scales. Resampling is done using pandas' `pd.core.frame.DataFrame.resample`; when downsampling, this effectively takes an average across the sampling period. For example, the combined OMNI-Wind data set is resampled on a 6-hour time cadence; a new sample at 12:00pm (noon) will be an average of values between 12:00pm and 6:00pm. The high-resolution magnetic field data from Wind is resampled into two separate data sets: one sampled at 10 samples per second (10 Hz), and one at one sample every 5 seconds (0.2 Hz); the 10 Hz data is used to compute Taylor scales as a higher resolution is needed, while the 0.2 Hz data is used to save computation time in computing the correlation scale as a lower resolution is sufficient. These scales are then calculated for each 6-hour interval of the respective data sets, to be matched to the OMNI-Wind merged data.

4.2.1 Autocorrelation function

A precursor for the calculations of both scales is the determining the autocorrelation function, defined again as Equation 4.1 [3], where $\mathbf{b}$ is a three-dimensional magnetic field time series, and r is the specified lag; in this case, a lag is one "shift" in the time series. The $\langle\cdot\rangle$ function denotes the correlation of a time series with itself when shifted by a given offset r. As the high-resolution `WI_H2_MFI` data provides magnetic field strength data in its three components, this computation can be done component-wise, then normalized as in Equation 4.2.

$$R(r) = \langle \mathbf{b}(x) \cdot \mathbf{b}(x+r) \rangle \tag{4.1}$$

$$R(r) = \frac{1}{3} \cdot \left(R_x(r) + R_y(r) + R_z(r) \right) \tag{4.2}$$

Autocorrelation functions $R(\cdot)$ are computed through statsmodels' `statsmodels.api.tsa.acf` function. The result is a list of autocorrelation values at the each of the number of lags specified. As estimations of the Taylor scale require a smaller portion of the autocorrelation function, 20 lags are calculated for this purpose. At 10 Hz, this translates to 2 seconds of lag. For estimating the correlation scale however, more lags are often required; 1500 lags are therefore used. at 0.2 Hz, this translates to 50 minutes of lag.

4.2.2 Calculating the Taylor microscale

For simplicity, the Taylor microscale is calculated as a time scale, as this process can be done independently from retrieving the data for solar wind speed. Values are therefore in seconds, and can be converted to a length scale through multiplication with flow speed. Two methods of calculating the Taylor scale were considered, both based on the theory provided in chapter 2.

Method 1: calculation through second derivative

Using the Taylor scale definition in Equation 4.3 and the fact that autocorrelation functions, when normalized, satisfy $R(0) = 1$, the Taylor scale can be estimated with a measure of the second derivative of the autocorrelation function at the origin, $f''(0)$.

$$\lambda_T^2 = \frac{f(0)}{f''(0)} \tag{4.3}$$

One method of determining $f''(0)$ is through directly retrieving it by differentiating the autocorrelation function twice. To achieve a continuous and (twice) differentiable autocorrelation function, a cubic spline can be fit to the discrete autocorrelation function; this is done with `scipy.interpolate.UnivariateSpline`. As seen in Figure 4.6, a cubic spline (orange curve) can model an autocorrelation function (blue curve) almost exactly given enough knots.

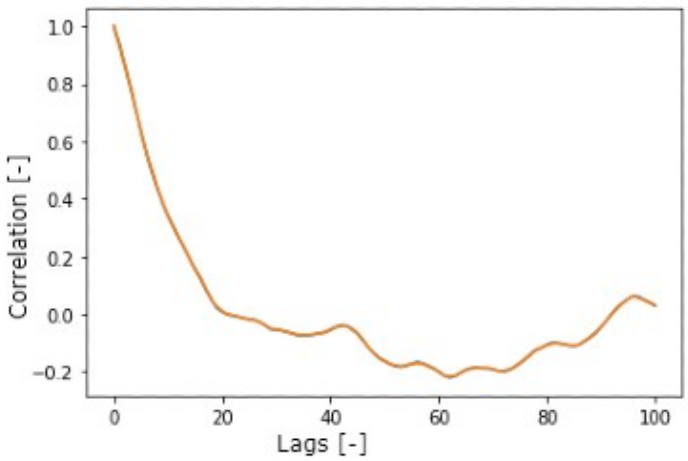


Figure 4.6: Example of a cubic spline fit (orange) to an autocorrelation function (blue)

As scipy's `UnivariateSpline` object provides a `.verbatim` method, the first and second derivative can be determined. The results from this can be compared to results from central finite differences from the discrete autocorrelation function, with a central difference defined as in Equation 4.4 with h being the step size. Results are plotted and compared in Figure 4.7.

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{4.4}$$

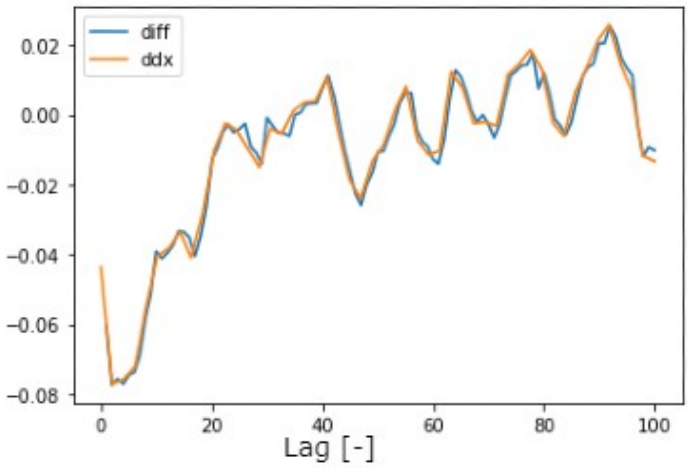


Figure 4.7: First derivative of cubic spline fit (orange) compared to finite difference of discrete autocorrelation function (blue)

The same process can then be repeated to arrive at the second derivative of the autocorrelation function, shown in Figure 4.8. Of note is that, since a central difference is used, the second derivative of the leading two values cannot be computed.

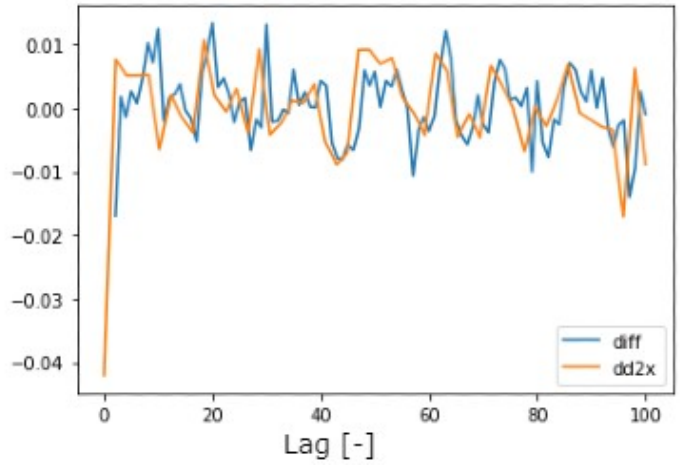


Figure 4.8: Second derivative of cubic spline fit (orange) compared to finite difference of discrete autocorrelation function (blue)

From Figure 4.7, both methods methods produce similar results, but results for the second derivative in Figure 4.8 differ slightly due to the discrete nature of the original autocorrelation function. Ultimately, the method of determining the second derivative directly was not used, partially due to it not being used in the literature, but also that results are very sensitive to behavior at the origin. In rare cases, the second derivative would result in a positive value due to small artifacts at the origin. The second method is more robust to these issues, and is supported by works in the literature.

Method 2: calculation through parabolic fit intercept

The method used in the literature to estimate the Taylor microscale is that of fitting a parabola near the origin of the autocorrelation function, then finding the x-intercept of the fitted curve, shown again in a sketch in Figure 4.9.

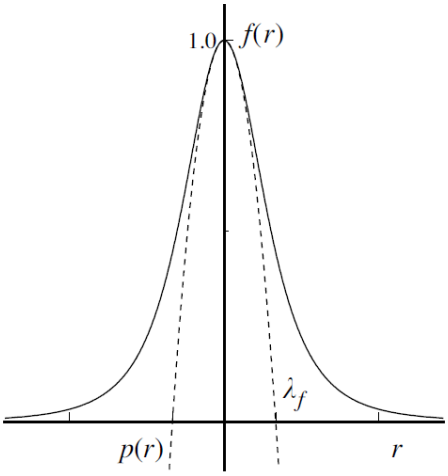


Figure 4.9: Sketch of autocorrelation function and parabola fit for finding the Taylor microscale [1]

The transverse Taylor scale is then found with Equation 4.5 [1]. $f''(0)$ is found through fitting an equation of the form $f(x) = a \cdot x^2 + 1$ near the origin, optimizing the a parameter. a is then related to the second derivative of the parabola as $f'(x) = 2a \cdot x$ and $f''(x) = 2a$. The number of points to fit the parabolic curve on is another important decision to make; this was decided through inspecting plots of different possible choices to be 2 seconds worth of lag.

$$x_{f(x)=0} = \lambda_T = \left(\frac{-f''(0)}{2} \right)^{-1/2} \tag{4.5}$$

4.2.3 Calculating the correlation scale

Two methods are also considered for calculating the correlation scale, both supported by and used in literature. As with the Taylor scale, these are calculated as time scales as the computations are done independently of the flow speed measurements. Again, a lower resolution of measurements is used for computation of the correlation scale, at 0.2 Hz up to 1500 lags. The two methods used are outlined below, and both are used to apply to the data in order to evaluate the differences later.

Method 1: calculation through an exponential fit

A function of the form $f(r) = \exp(-r/\lambda_c)$ is fit to a specified section of the autocorrelation function through optimizing the λ_c parameter. The resulting optimal value of λ_c is then an estimation of the correlation scale; this is commonly used in literature [3]. Note that this is the same autocorrelation function used to calculate the Taylor scale, but the function is fit on a far larger portion, which resembles an exponential decay shape; the fit for the Taylor scale focuses on a very local parabolic behavior near the origin. The extent of the autocorrelation function to fit on was determined to be 1000 seconds worth of lag, through some trial and error and inspection of plotted results. Examples of fits are shown and compared to the other method in Figure 4.11 and Figure 4.12.

Method 2: estimation through 1/e trick

This method employs a trick that essentially states that "if the autocorrelation function followed a perfect exponential decay function as specified by $f(r) = \exp(-r/\lambda_c)$, then the value for λ_c is simply the value for r at which $f(r) = 1/e$," providing a computationally cheap method of estimating the correlation scale. This can be confirmed mathematically when considering r_c as the value for r when $f(r) = 1/e$ through the following: $\exp(-r_c/\lambda_c) = e^{-1}$, and therefore $r_c/\lambda_c = 1$ and $r_c = \lambda_c$.

Naturally, the value for r_c using this method will rarely fall on one of the specified lags, but rather somewhere between two lags; as these intervals are quite small ($\approx 5sec$), a standard linear interpolation is used. This is shown in a sketch in Equation 4.6, with parameters defined in Figure 4.10 where x_opt is the desired r_c. An algorithm simply searches an array of autocorrelation values until a value under $1/e$ is encountered, representing the point (x_2, y_2); (x_1, y_1) is then the preceding point.

$$\begin{aligned} \frac{y_1 - 1/e}{y_1 - y_2} &= \frac{x_{opt} - x_1}{x_2 - x_1} \\ x_{opt} &= x_1 + \left(\frac{y_1 - 1/e}{y_1 - y_2}\right)(x_2 - x_1) \end{aligned} \tag{4.6}$$

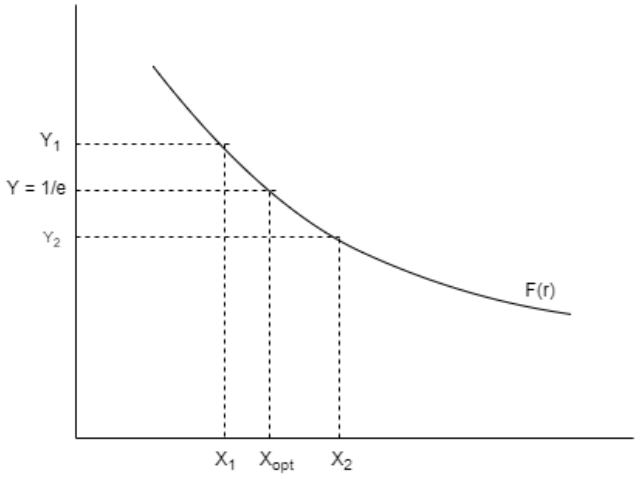


Figure 4.10: Sketch of linear interpolation process for estimating the correlation scale through the 1/e trick. The point (x_2, y_2) is the first point where y is less than $1/e$, and (x_1, y_1) is the preceding point. (x_2, y_2) is simply found through a sequential search of autocorrelation values. X_{opt} can then be calculated with Equation 4.6 as the estimate for the correlation scale.

Two examples of results from both methods are shown and plotted in Figure 4.11 and Figure 4.12. Figure 4.11 shows a clean exponential fit, meaning the 1/e trick results in a similar value. Naturally, most fits will not be as neat, as shown in Figure 4.12, where the results of both methods differ slightly. This difference is considered acceptable as both are merely estimates of large-scale, fairly abstract phenomena. Again, results from both methods are kept for further analysis.

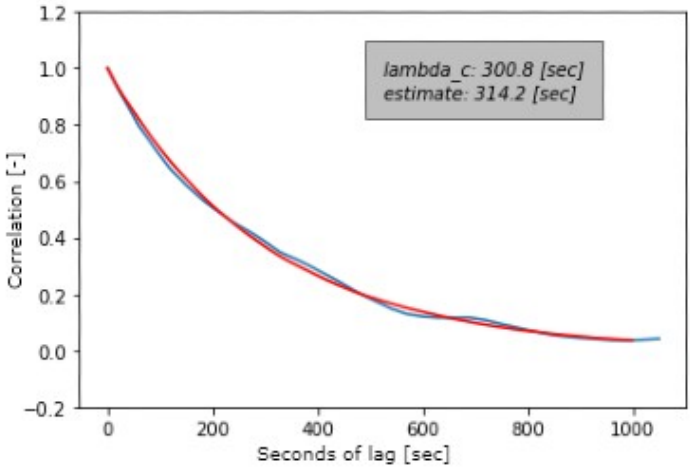


Figure 4.11: Example 1 of using both methods to compute the correlation scale. An accurate fit is made (red) and the 1/e trick therefore results in a very similar value

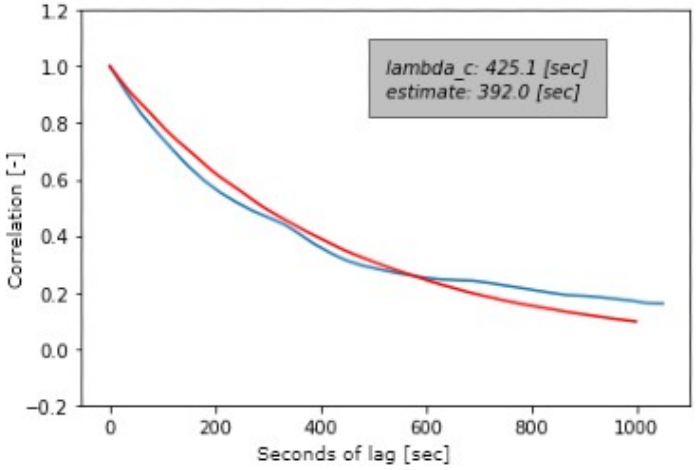


Figure 4.12: Example 2 of using both methods to compute the correlation scale. The fit (red) is not as clean as the autocorrelation function (blue) does not exhibit perfectly exponential behavior. In cases like these, the 1/e method will differ from the fit value slightly. These margins are considered acceptable, however, and are discussed later

4.2.4 Data combination

With the three data set components collected (averaged properties from a merged OMNI-Wind data set, calculated correlation scales from Wind data, and calculated Taylor scales using high resolution Wind data), they can be merged by timestamp into a single data set. This is done using `pandas.merge` on the timestamp column, matching up the data instances. Through processing 5 years of data (2016-2020) from the specified data sources, a final data set of size 6908×20 is created. Missing entries result from any of the following:

- Greater than 40% of time series data was missing. In such a case, the interval is discarded.
- The associated CDF file is corrupted or ran into an exception when being read by `cdflib`. This was the most common of the three occurrences and accounted for the majority of data loss.

- A fit was not successful in the calculation of the Taylor or correlation scale; in some rare cases the `scipy.optimize import curve\_fit` method used to optimize the respective parameters will fail to converge. In such a case, the methods for calculating both scales will discard the instance.

The overall data set creation pipeline is summarized in Figure 4.13, where raw data from Wind and OMNI missions are processed as shown. Namely, 6-hour averages are extracted from `OMNI_HRO2_5MIN`, `WI_ELM2_3DP`, and `WI_PLSP_3DP`. Taylor scales, correlation scale, and RMS magnetic field strengths are extracted from 6-hour intervals of `WI_H2_MFI`. Instances are then merged on their timestamps to arrive at the final set of data.

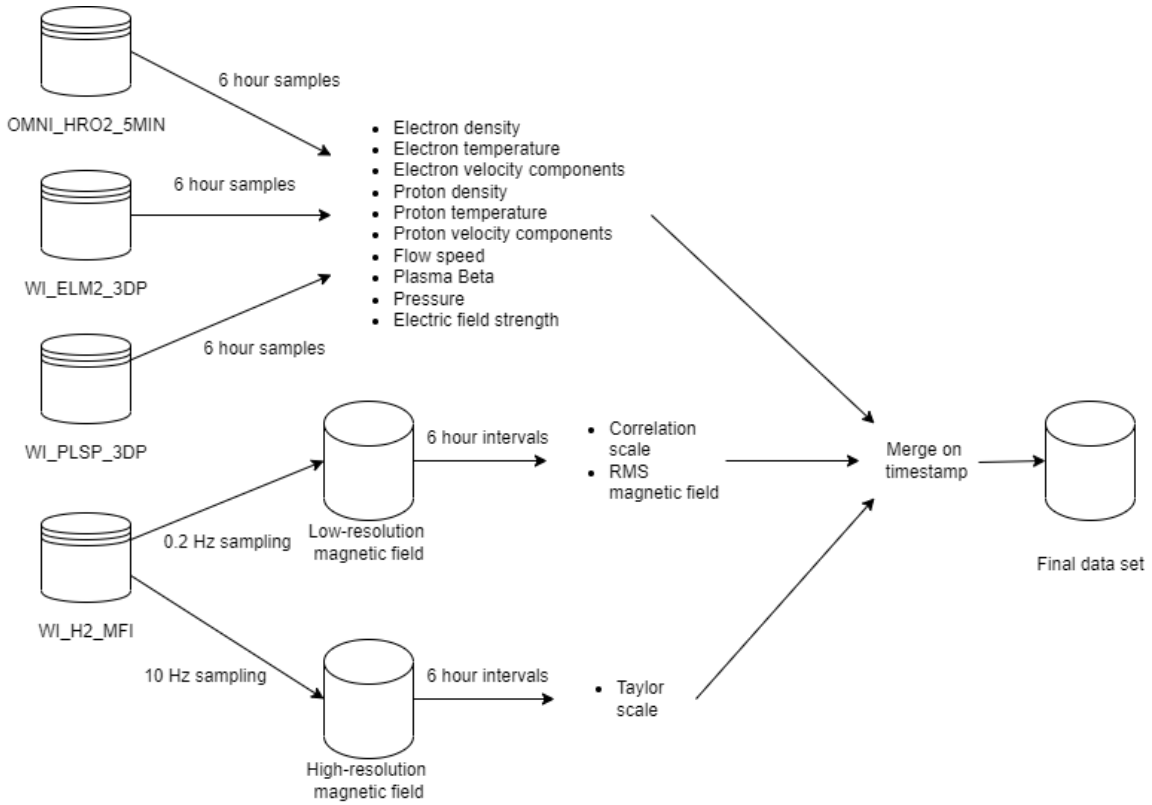


Figure 4.13: Summary of data creation pipeline. OMNI and Wind data sets are combined for large-scale averages of various flow properties. High resolution Wind data is resampled into two data sets, one high resolution data set for calculating the Taylor scale and one lower resolution for calculating the correlation scale and RMS magnetic field strength. The resulting instances can be matched up on their timestamps.

Chapter 5

Exploratory data analysis

Exploratory Data Analysis (EDA) is a process in which data is investigated and visualized to uncover potential intricacies and patterns that may aid in following analysis/machine learning steps. Often times simply visualizing data before designing and applying machine learning algorithms can be beneficial as discoveries made during EDA can inform design choices. While EDA is designed to be an open-ended endeavor, a standard approach is to explore the following [45]:

- Data types present in the data; this can also often be informed by some prior domain knowledge. A distinction can be made between data types of variables (numerical, categorical, ordinal, nominal, etc.) and purpose of variables (target variable, explanatory variable, metadata, etc.) within the data set.
- Outliers or other oddities within the data; this is also often informed through domain knowledge.
- Data visualizations; these are often rough plots that offer insight into patterns present in the data. Options for visualizations differ for different data types, with the purpose of exploring aspects like trends, correlations, and distributions.

The data used in this work is fairly simple in the sense that all features are numerical and continuous explanatory variables. EDA also partially achieves the goals of this work since it inherently provides insight into the potential relationships between the explanatory variables and the target variable. The remainder of the chapter is split into high-level exploration, which aims to investigate and visualize the data as a whole, and more targeted exploration, where more specific questions are treated. Finally, the possibility of sparsely representing the given data is investigated using dimensionality reduction techniques.

5.1 High-level exploration

High level exploration is begun by simply observing slices of data, as the ordered data is essentially a time series with samples every six hours, disregarding discarded instances. Such a data slice is shown in Figure 5.1, taken from approximately a 40-day period from January 1st 2016. Some variables exhibit much lower frequency fluctuations than others, namely the first electron velocity component, proton temperature, and flow speed. The remaining features tend to fluctuate more sporadically. The three calculated features (correlation scale through two methods and Taylor scale) also fluctuate quite sporadically. The peaks in these three calculated features seem to coincide with one each other, as well as features like the proton density and electron density; this indicates a possible connection. As stated in chapter 4, some redundancy was built into the data set to account for possible differences in the combined data sources. Namely, two measures for proton density ("Proton_density"

and “Proton_density2”) and proton temperature (“Proton_temp” and “Temperature”) exist. From observations such as that shown in Figure 5.1, it is clear that both sources capture the same information in these fields, though at a different scale in the case of the proton temperature. The results for both methods of estimating the correlation scale are still present, and will be investigated later in the chapter.

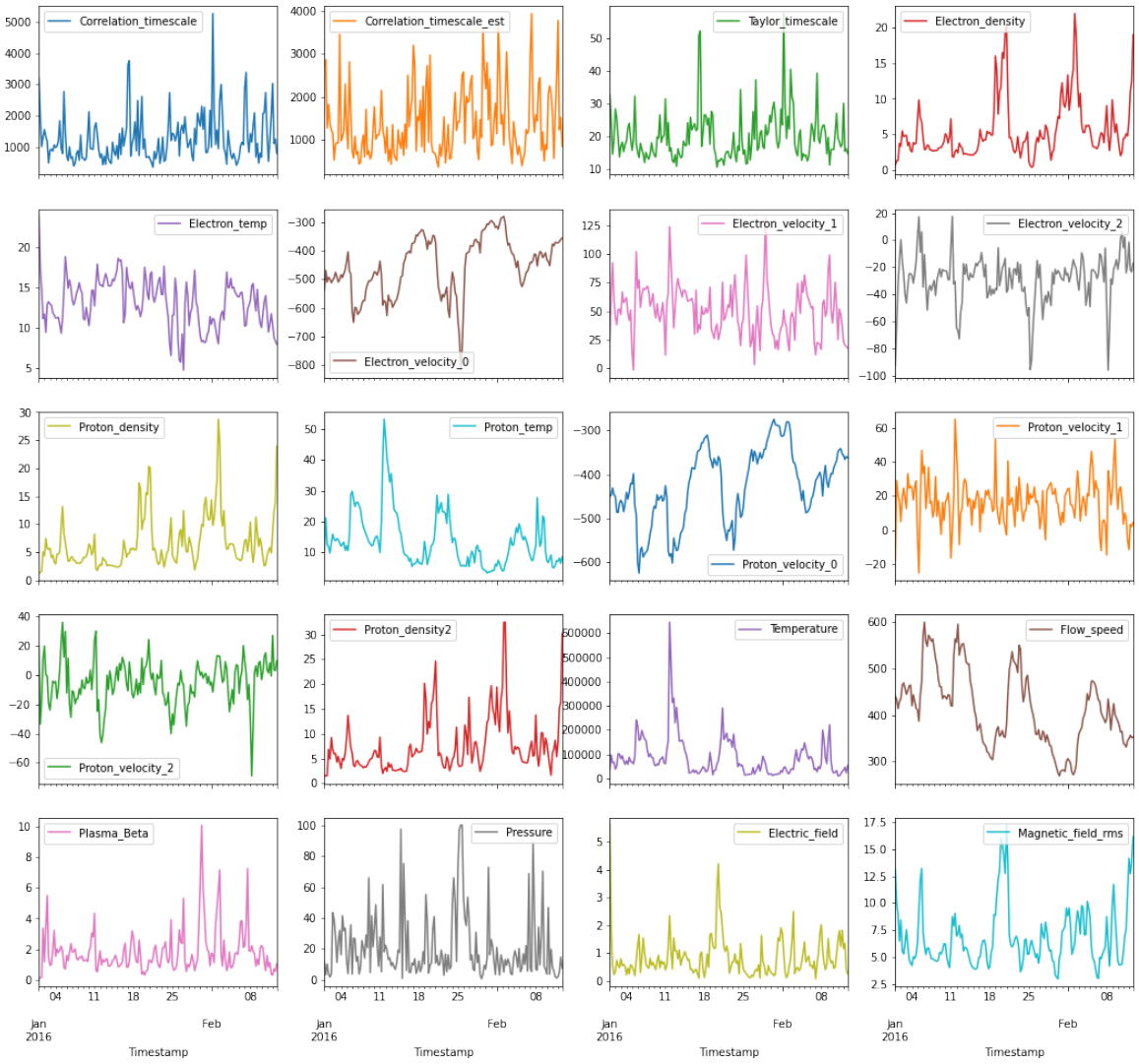


Figure 5.1: High-level view of a section of the data set in a time-series format. Approximately 170 instances are displayed, equating to around 40 days. Of note is that the computed features (correlation and Taylor scales) exhibit a large amount of fluctuations, much like some other features and unlike some others. As a general observation, some peaks visible in the correlation scale and Taylor scale time series seem to loosely coincide with peaks in other features such as the proton density, electron density, and each other. The data redundancy added for exploratory purposes seem to be unnecessary as both proton temperature measures appear to capture the same data, as well as the temperature and proton temperature measures.

Next, each the distributions of each feature can be explored through boxplots and histograms. Boxplots help visualize the distribution of data points within a feature through showing the interquartile range (middle 50% of values) and extreme values of data. His-

tograms also help visualize distributions through grouping into value bins and plotting data frequencies in these bins; histograms in this section are built using 60 bins.

Figure 5.2 and Figure 5.3 show the box plots and histograms for the calculated time scales, respectively. All three seem to have extended right tails, but follow a general Gaussian shape. Moreover, the distributions of the correlation scale (through the fit method) and the Taylor scale follow a similar shape to each other. The two methods of calculating the correlation scale yield results with similar shapes, though the interquartile range of results from the 1/e computation method is significantly larger that of the fit method. These two methods are further compared later.

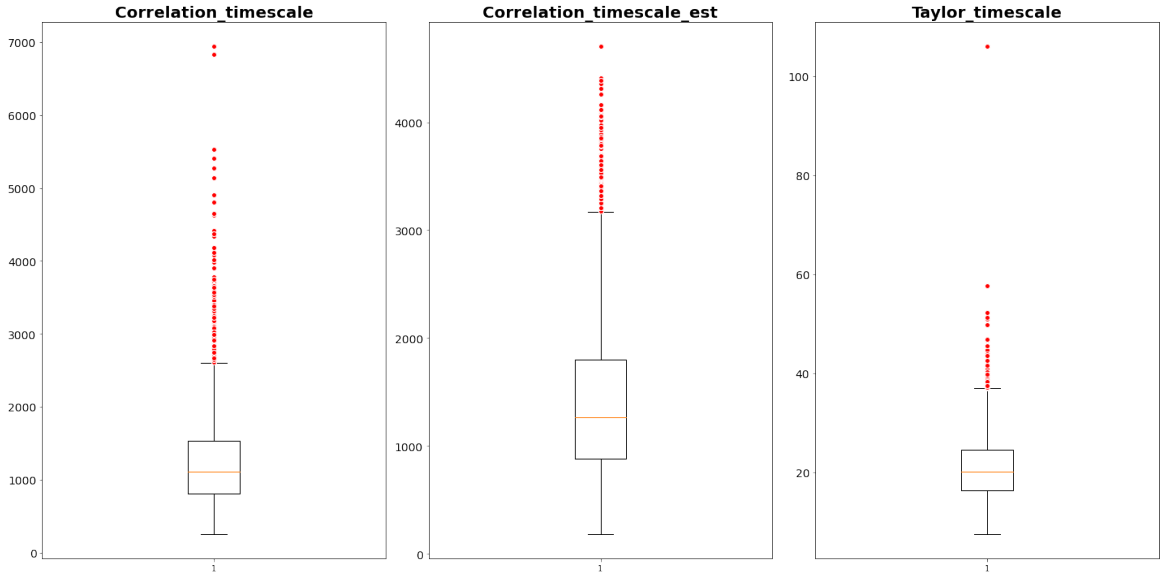


Figure 5.2: Box plots of the computed features: correlation scale (fit and 1/e method) and Taylor scale. A fairly large proportion of data points appear outside of the interquartile range (red points), specifically above the 75th percentile.

Figure 5.4 and Figure 5.5 show box plots and histograms of the electron-related features. Some features such as temperature and the last two velocity components are already fairly Gaussian shaped, while others are more skewed; these features can be standardized in a preprocessing stage before machine learning techniques are applied. More domain knowledge is needed to evaluate whether the distributions of the observations are typical within the 5-year observation period.

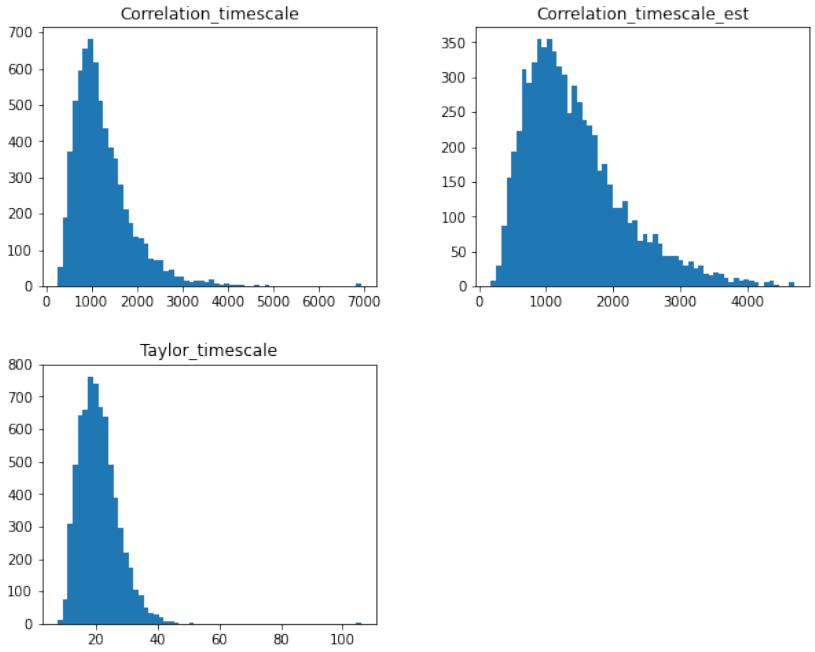


Figure 5.3: Histogram of the computed features: correlation scale (fit and 1/e method) and Taylor scale. Distributions follow a loosely Gaussian shape, with longer tails to the right. Distributions of both correlation scale results follow a similar shape, though results from the 1/e method ("Correlation_timescale_est") is flatter than those of the fit method ("Correlation_timescale."

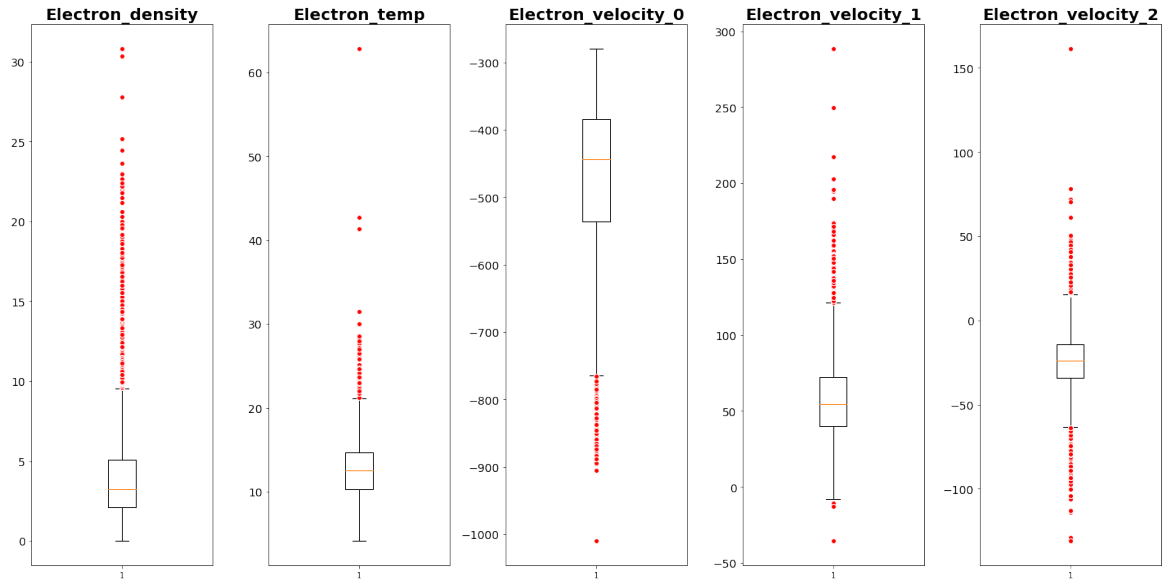


Figure 5.4: Box plots of electron-related features: density, temperature, and components of velocity. Electron density and the first component of electron velocity exhibit a heavily skewed distribution with a large proportion of outlier points (red points)

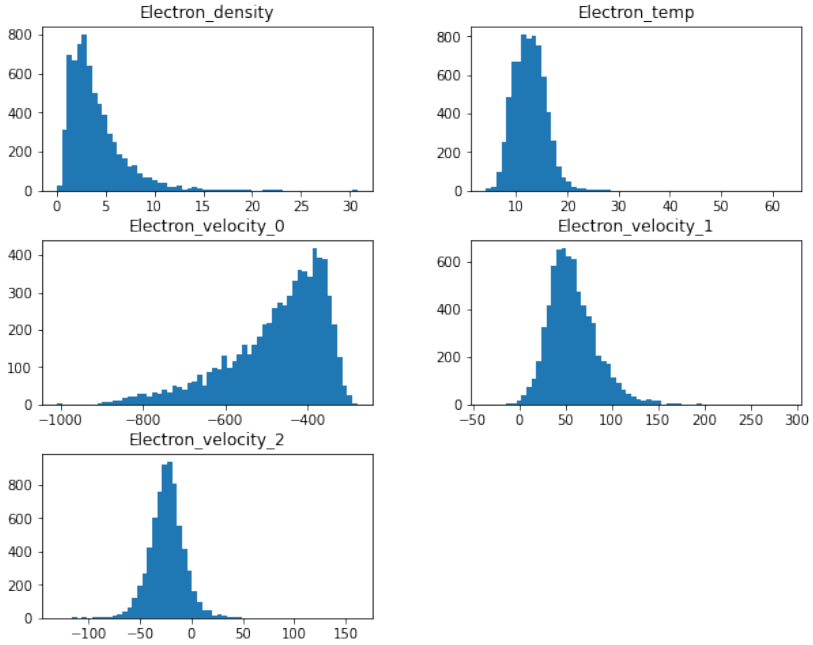


Figure 5.5: Histogram of electron-related features: density, temperature, and components of velocity. Electron density and the first component of electron velocity exhibit a heavily skewed distribution to the right and left, respectively. The other features have fairly Gaussian distributions.

Figure 5.6 and Figure 5.7 show box plots and histograms of the proton-related features. Again, some features such as the last two velocity components are already fairly Gaussian shaped, while the others more skewed left or right, indicating that standardization may be useful. Similar to the electron features, the first GSE component of velocity seems to have a minimum (magnitude) velocity, causing the distribution to be skewed. Naturally, temperature and density is an absolute measure, which may contribute to more values being concentrated around zero. The other two components being distributed symmetrically near zero almost acts as noise when combined with the more dominant first component. This behavior in both the electron and proton velocities may indicate that the first component is the most important to consider for machine learning purposes.

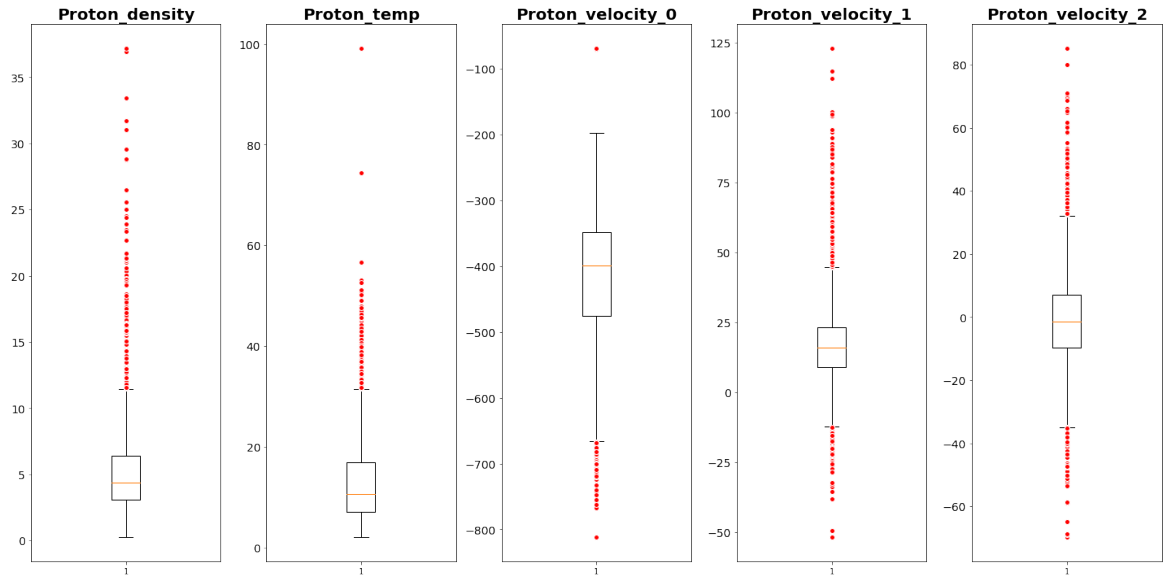


Figure 5.6: Box plots of proton-related features: density, temperature, and components of velocity. The last two velocity components exhibit a fairly centered distribution, while other features are more skewed.

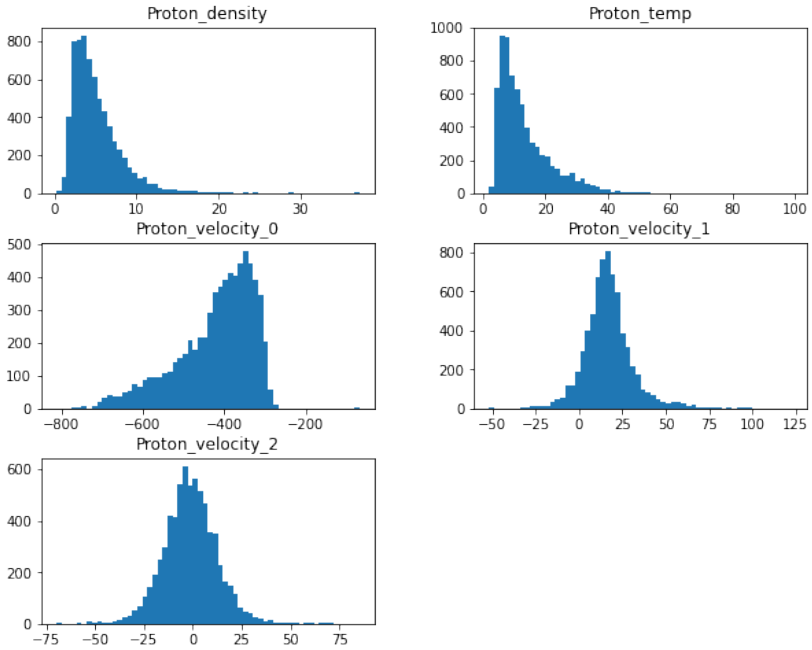


Figure 5.7: Histogram of proton-related features: density, temperature, and components of velocity. The last two velocity component have distributions that resemble a Gaussian shape. Proton temperature and density are heavily skewed.

Finally, the box plots and histograms of the remaining features are shown in Figure 5.8 and Figure 5.9. All features are absolute measures and are therefore all positive. All features have fairly skewed distributions and should be standardized in preprocessing.

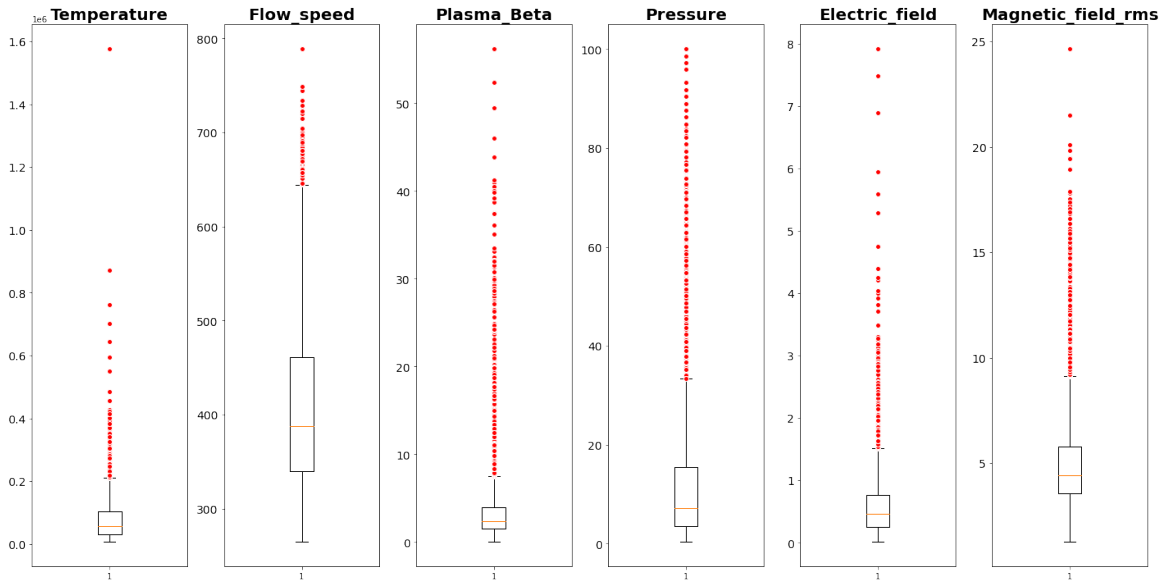


Figure 5.8: Box plots of remaining features: proton temperature, flow speed, plasma beta, pressure, electric field strength, and magnetic field RMS. All features contain a large proportion of outlier points (red points).

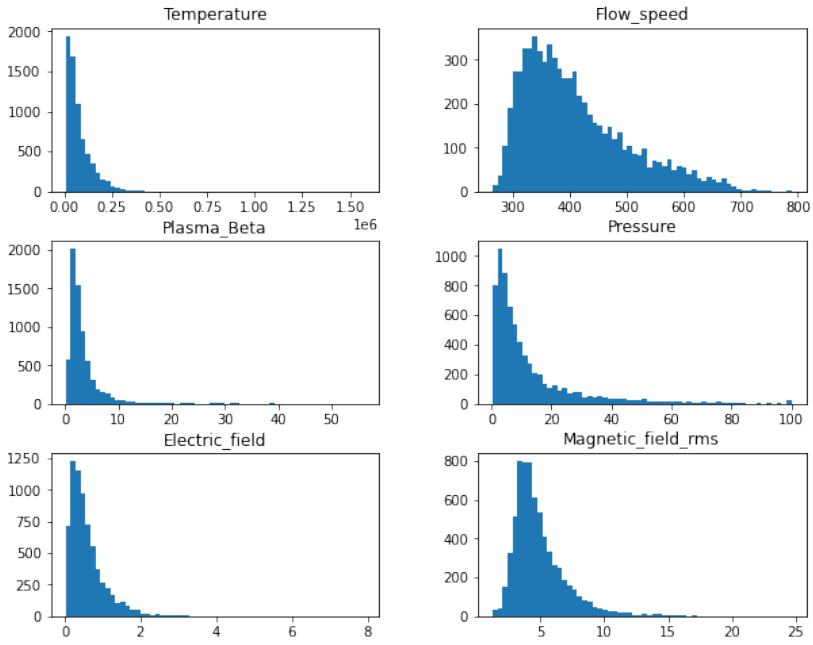


Figure 5.9: Histograms of remaining features: proton temperature, flow speed, plasma beta, pressure, electric field strength, and magnetic field RMS. All features exhibit skewed distributions.

5.2 Targeted exploration

Next, some more targeted exploration can be done, investigating the following questions:

- Data validation: does the collected magnetic field data behave as expected according to known power law relationships?
- How do the two methods of computing the correlation scale compare?
- What are the correlations between all of the collected features?
- What is the nature of the apparent correlation between the correlation scale and Taylor scale observed previously?

Data verification and validation is a crucial step in EDA, ensuring that the collected data behaves as expected and thus were collected correctly. One way of validating the collected magnetic field strength data is to observe if the -5/3 power law is visible within a power density spectrum; the result can also be compared to those from available literature. Figure 5.10 and Figure 5.11 show power spectra constructed from approximately 7 hours of magnetic field strength measurements from January 1st 2016, using the low-resolution and high-resolution data set respectively. In both figures, a floating -5/3 power law if plotted to help verify if the inertial scale power law relationships is observed. Typically, such power density spectra are put through some smoothing processes to remove the noisy areas at higher frequencies. This noise is more noticeable in the spectrum made using high-resolution data in Figure 5.11. Nonetheless, both spectra exhibit the expected -5/3 power law.

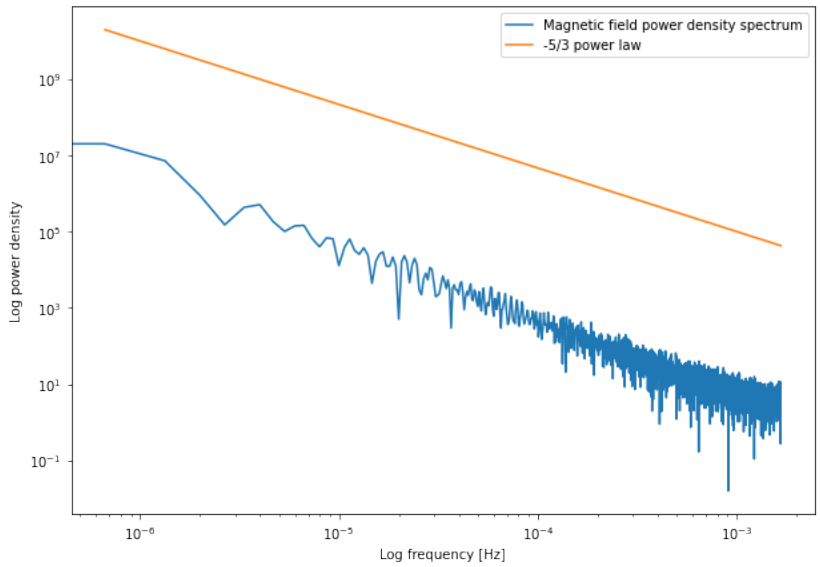


Figure 5.10: Example of raw power density spectrum computation of magnetic field strength data from 7 hours of the low-resolution (0.2 Hz) data from January 1st 2016. A floating -5/3 power law fit is also shown to confirm the power law behavior of the inertial range is captured.

Moreover, these results can be compared to those found in the literature. Figure 5.12 shows multiple power density spectra traces from both fast and slow wind, computed from both the Ulysses and Helios 2 spacecraft at different distances. The form of the spectra follows the results shown in Figure 5.10. Figure 5.11 shows a more noisy spectrum, more comparable to that shown in Figure 5.13 [3]; here, the same -5/3 power law is observed in the inertial range, but another -8/3 power law is visible at higher frequencies. This change in power law occurs at a so-called spectral break-point [1] [2], though the shape in this region

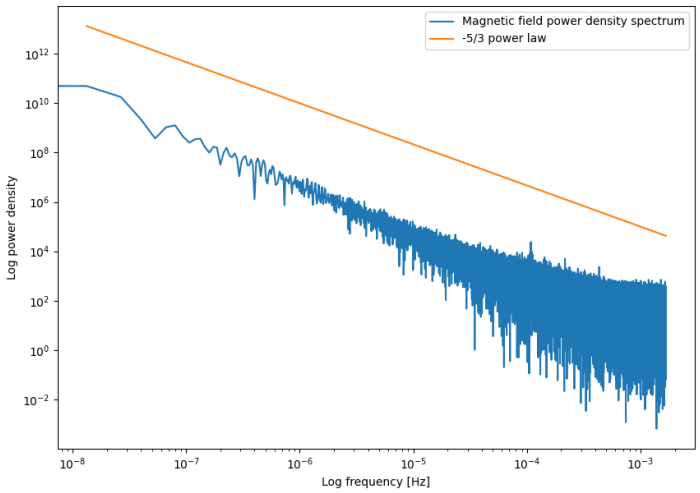


Figure 5.11: Example of raw power density spectrum computation of magnetic field strength data from 7 hours of the low-resolution (0.2 Hz) data from January 1st 2016. A floating -5/3 power law fit is also shown to confirm the power law behavior of the inertial range is captured.

is often subject to a high level of noise. In Figure 5.13, the faded region in the spectrum represents the area in which the signal-to-noise ratio falls below 5 [3]. Though, some further smoothing is needed to discover if the change in power law is visible in Figure 5.13. Through computing these power density spectra and comparing them with those found in literature, the validity of the collected data is more certain.

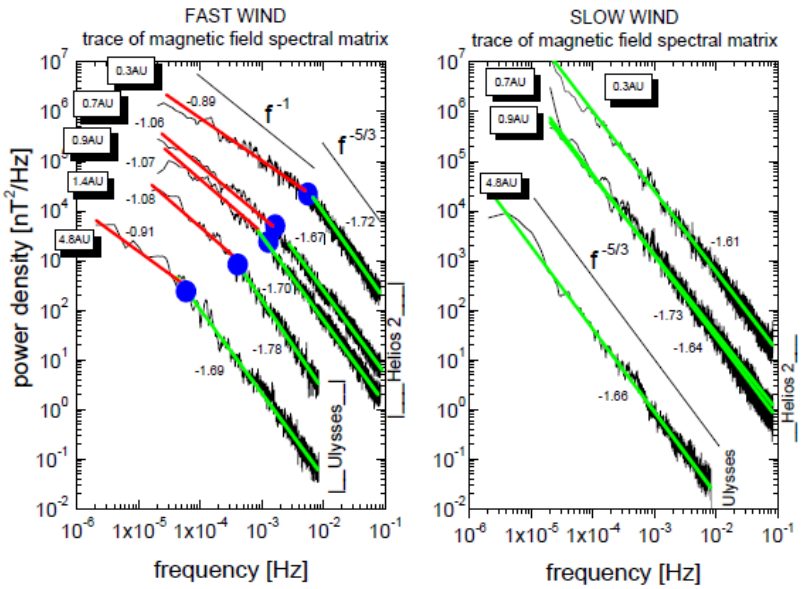


Figure 5.12: Examples of traces of magnetic field power density spectra, computed from both the Ulysses and Helios 2 spacecraft at different distances [2]

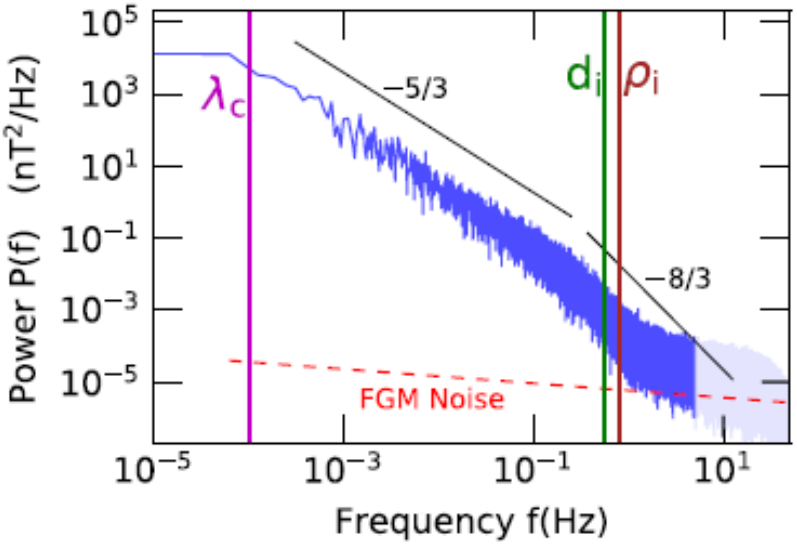


Figure 5.13: Example of power density spectrum with a spectral break point visible where the power law changes. The faded region is where the signal-to-noise ratio falls below 5 [3]

In a similar way, the rough order of magnitude of the computed scales can also be compared to those found in literature. The average computed Taylor scale in the data set is 20.03 seconds, and the average computed correlation scale is 1197 seconds; for this analysis, the results from the fit method are used as this method is also used in the literature [3]. To convert to a rough average length scale, the time scales can be multiplied by the average flow speed in the data set, 410 km/s.

Bandyopadhyay et al. [3] observes a Taylor scale of $\lambda_T = 6933km$ and a correlation length of $\lambda_c = 320000km$. Matthaeus et al. [16] cites an estimate of the Taylor scale $\lambda_T = 2400 \pm 100km$, and observes values ranging from approximately 1000 to 5000 km.

Using the averages found in the data set, a length scale average can be computed as $\lambda_T \approx 8200km$ and $\lambda_c \approx 490000km$. As estimates for these scales vary to a small degree depending on the context in which they were taken, the estimates observed in the data set are considered in an acceptable range, also given that studies of turbulent length scales often regard order-of-magnitude estimations of such measures [1].

Next, the results for both correlation scale estimation methods can be compared. Figure 5.14 shows a histogram comparison of values; while both distributions take a similar shape, the 1/e estimation method returns more larger values than the fit method. This is again visible in a hexagonal bin two-dimensional histogram heatmap plot in Figure 5.15. Here, the red diagonal represents the line where both methods return the same value. More values lie above this line than below, indicating again that the 1/e method produces larger results. Interestingly, the two methods agree more when correlation lengths are small, seem through the near-linear behavior near the origin containing a high density of values; after this small region, values diverge. A possible explanation for this comes with the fact that autocorrelation functions were seen to "flatten out" more after the region in which the exponential function would be fit; a flatter function would cause the 1/e method possibly overshoot the value that would otherwise match that from the fit method if the function observed perfect exponential behavior.

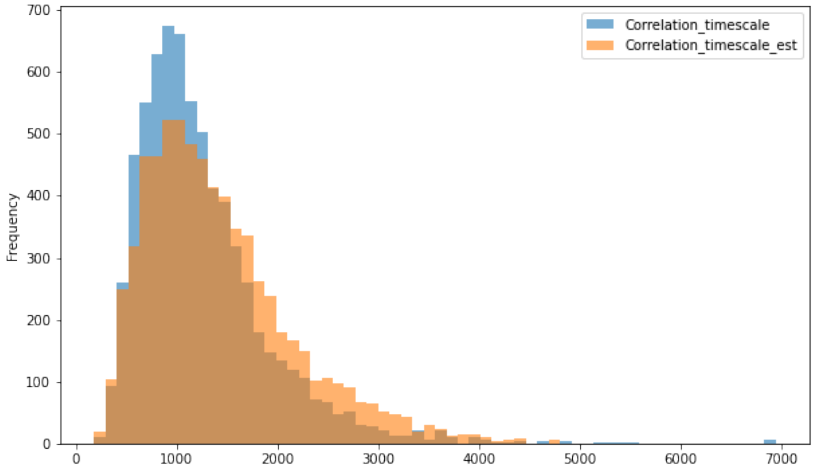


Figure 5.14: Histogram showing the correlation scale estimates from the fit method (blue) and the 1/e method (orange). Distributions observe a similar shape, with that of the 1/e method being more flattened towards the right.

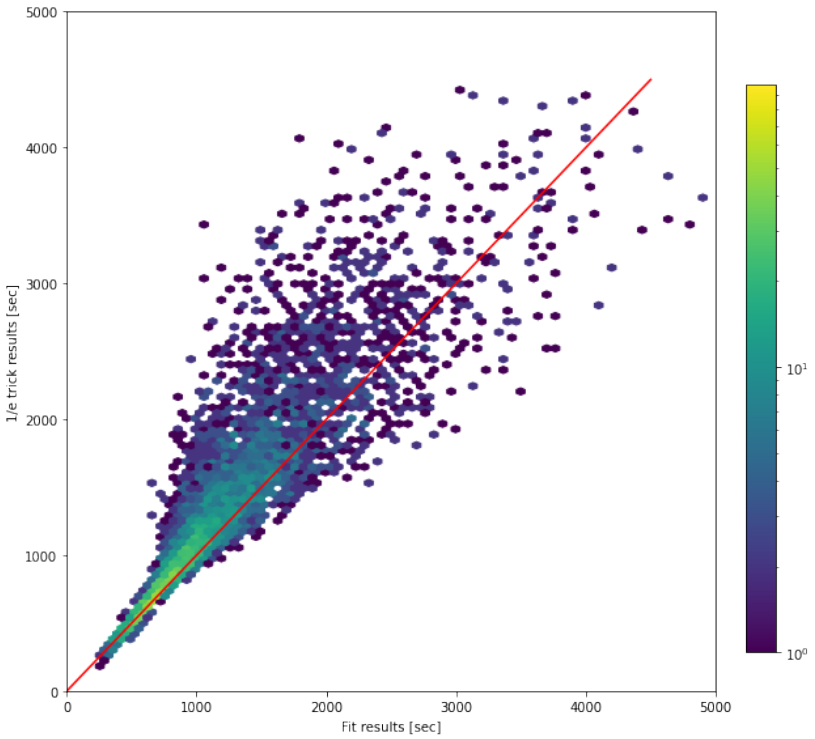


Figure 5.15: A two-dimensional, hexagonally binned histogram heatmap with low values whited out. The red line shows the cases in which results from both methods would be equal. More values lie above this line than below, indicating again that the 1/e method produces larger results overall. Additionally, the two methods tend to agree more near the origin, where points are near linear; the density of points in this area is also high.

A broad view of feature correlations can be obtained from a correlation heatmap as shown in Figure 5.16. Here, pairwise correlations are computed with the Pearson's correlation coefficient, which ranges from -1 to 1 where values closer to the limits indicate more negatively or positively correlated variables, respectively. Cells are colored according to

their values, with values close to zero being colored white and more correlated values colored either blue or brown. Such a heatmap can both verify known relationships and uncover potential new ones. Some notable observations include the following:

- Correlation scale and Taylor scale appear to be strongly correlated.
- Both correlation scale computation methods have similar correlations with other variables (visible from the first two rows), supported by the observations made previously when comparing the two.
- The correlations with the Taylor scale (third row) are of particular interest as this is the target variable.
- Electron-related features are correlated in expected ways, noting that the first GSE component of velocity is negative.
- Pro-related features are correlated in expected ways, noting that the first GSE component of velocity is negative.
- Both proton temperature measures exhibit similar correlations, indicating that they contain the same information, supported by observations and comments made previously regarding redundancy. The same comment can be made with both proton temperature measures ("Proton_temp" and "Temperature").
- The last component of electron and proton velocity have very weak to no correlations with other variables, except to each other.
- Pressure has weak to no correlation to other variables.
- Flow speed expectedly is linked to electron and proton velocities, noting again that the first component of the GSE velocity is negative and flow speed is an absolute measure, explaining the negative Pearson coefficient.
- The strongest correlations besides those associated with flow speed seem to link the electron and proton values. An example is the apparent link between electron velocity and proton temperature. More domain knowledge is required to comment further on these supposed links.

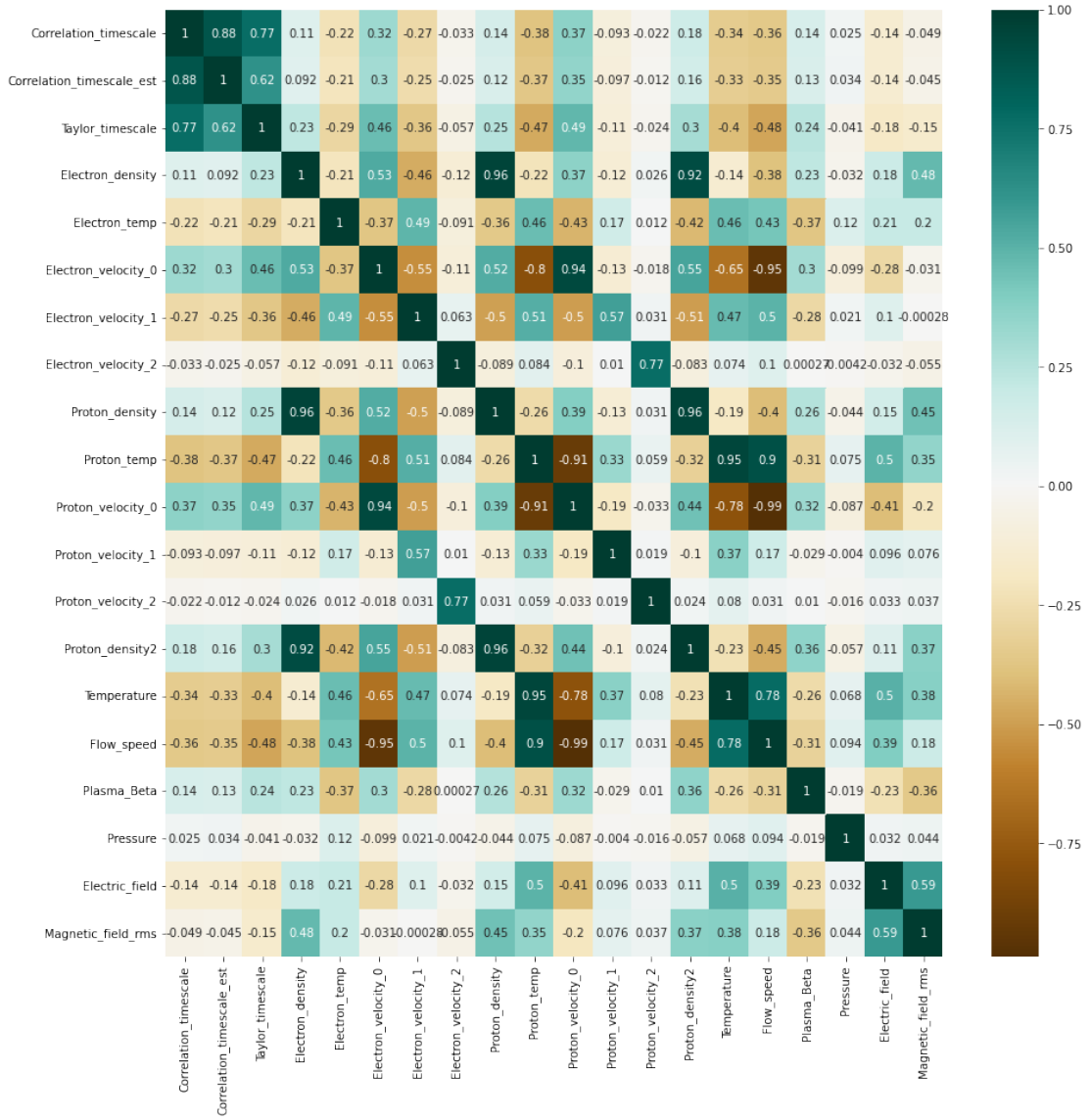


Figure 5.16: Correlation matrix of the full set of features. The Pearson correlation measure is shown for each pair of features. Cells are colored to denote magnitude and sign of correlation; pairs with a cell color closer to white are more loosely correlation. Of particular importance is the row/column of the Taylor time scale as this is the target variable.

The observed correlation between the correlation scale and the Taylor scale is of particular interest; this could be a true correlation, i.e. scales tend to increase with each other in solar wind, or is a consequence of how both quantities are computed. Naturally, both quantities have a relationship with the autocorrelation of magnetic field strength, so in this sense the correlation is somewhat expected; still, this is an interesting observation to further analyze. Figure 5.17 and Figure 5.18 show a comparison between the Taylor scale and the correlation scale from the fit method and 1/e method, respectively. A slice of approximately 80 hours of data is shown in these figures to illustrate the apparent links between the quantities. As both features exist on different scales, they are first rescaled between 0 and 1. The residual (difference) between the two values is also plotted in green. Note that this is not suitable for proper statistical analysis, but is done to more effectively visualize these correlations. From Figure 5.17, many spikes in values almost exactly coincide. As a

general observation, the values also seem to covary, smaller local peaks often also seem to line up. Residuals also naturally stay fairly low. Figure 5.18 shows similar behavior; some large peaks also coincide, but both plots are less similar overall. The correlation scale using this method tends to fluctuate more drastically, reflected in the higher residuals overall.

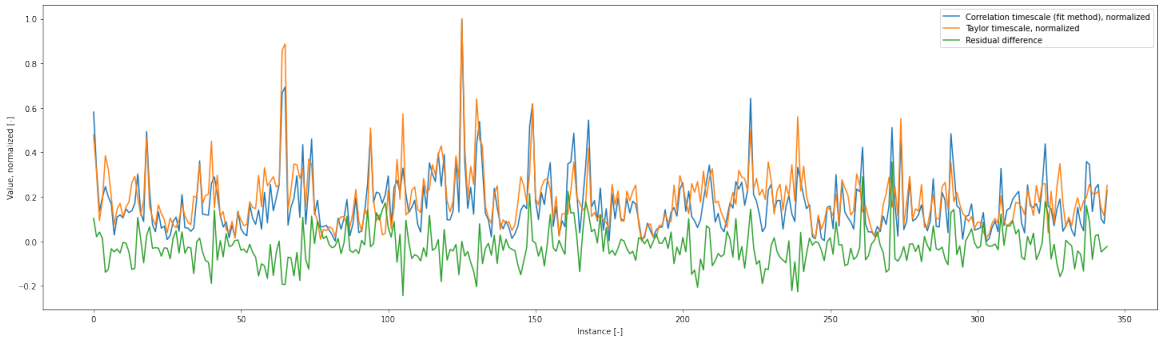


Figure 5.17: A comparison between the Taylor scale and the correlation scale computed through the fit method. Values from both features are min-max normalize to vary between 0 and 1, and a residual (difference) is also plotted in green to aid in visualizing the difference. Large peaks seem to coincide almost exactly. In a global sense, the features seem to covary, which is supported by Figure 5.16.

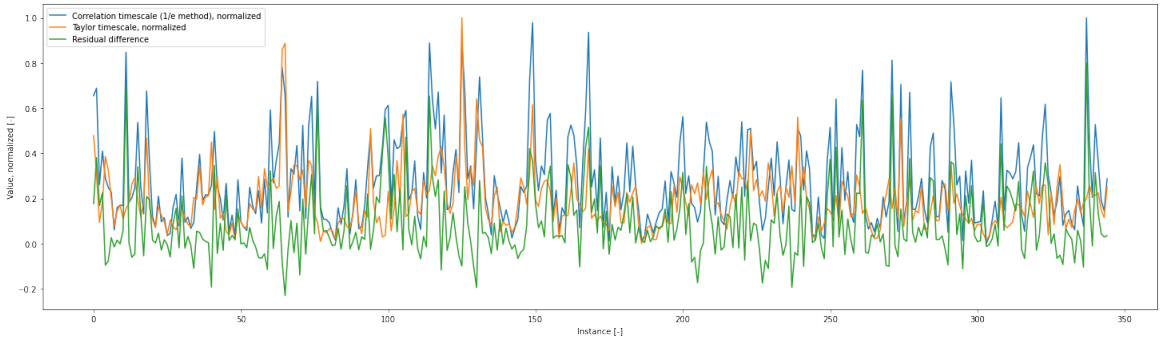


Figure 5.18: A comparison between the Taylor scale and the correlation scale computed through the 1/e method. Values from both features are min-max normalize to vary between 0 and 1, and a residual (difference) is also plotted in green to aid in visualizing the difference. Large peaks tend to coincide like in Figure 5.17, but the overall curves seem to match to a lesser degree. Values for the correlation scale seem to fluctuate more sporadically, creating more large peaks than in Figure 5.17.

The overall distribution of Taylor-correlation scale pairs is also of interest; hexagonal bin two-dimensional histograms are shown in Figure 5.19 and Figure 5.20 for both methods respectively. From both plots, a fairly standard convex-shaped distribution is observed, with a dense area of points that diminishes radially outwards. The distribution for the 1/e method is once again seen to be more spread out than that of the fit method.

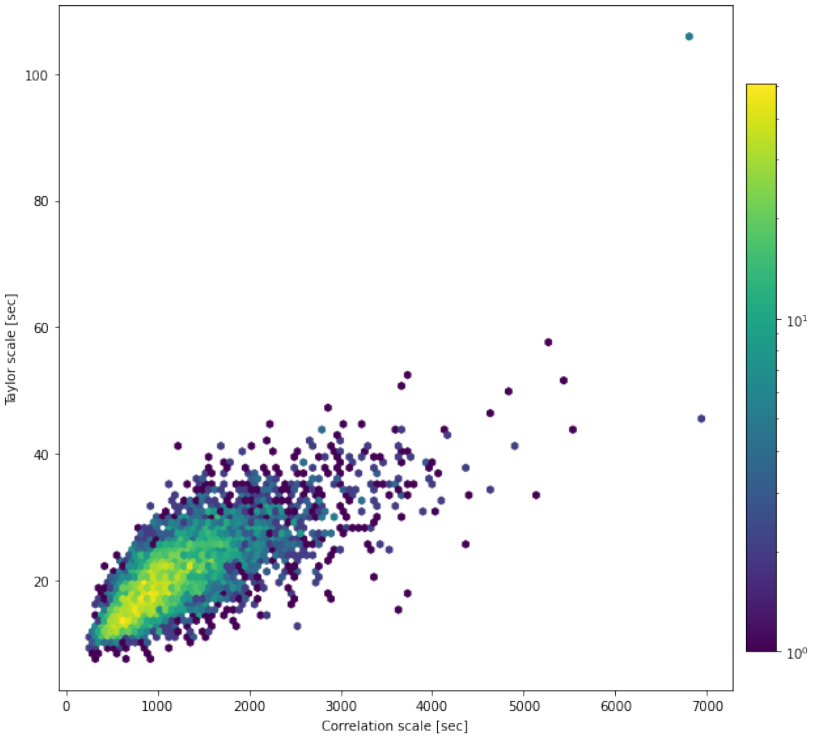


Figure 5.19: Hexagonal 2D-histogram heatmap of Taylor scale and correlation scale values computed with the fit method. Areas without data in the heatmap are whited out.

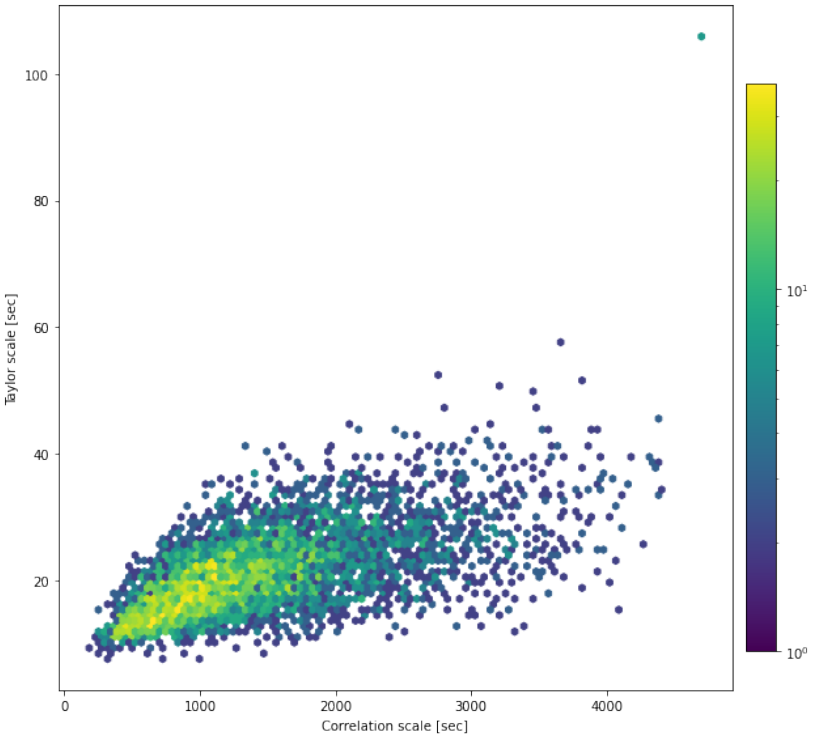


Figure 5.20: Hexagonal 2D-histogram heatmap of Taylor scale and correlation scale values computed with the 1/e method. Areas without data in the heatmap are whited out.

5.3 Sparse data representation and dimensionality reduction

Dimensionality reduction techniques can help uncover possible sparse representations of the data, and is often used in EDA or prior to machine learning as shown in chapter 2. Principal Component Analysis (PCA) is one of such techniques, which constructs a new set of orthogonal principal axes along directions of maximum data variance; the result is a lower-dimensional representation of the same data using linear combinations of the original features as principal axes. Investigating sparse representation is closely coupled with evaluating redundant or irrelevant features; the various low correlations seen in Figure 5.16 suggests that the data can be represented more concisely towards the goal of predicting the Taylor scale. PCA is performed using five principal components, with the results shown in Table 5.1; here, the top five ranking (highest eigenvalue) principal axes are shown with their associated explained variance. Note that the target variable, the Taylor scale, is excluded when performing PCA. From the table, it is evident that the input features can be sparely represented with even two principal components, which points to the viability of performing feature selection with machine learning to uncover deeper relationships. This is not unlike the results shown in [10], where spectra can be sparsely represented with only a few principal components.

Table 5.1: Results of Principal Component Analysis with five principal components. The first principal component already explains the majority of data variance.

Principal component number	**Explained variance ratio**	**Cumulative explained variance**
1	0.999822	0.999822
2	0.000163	0.999985
3	1.172563e-05	0.999997
4	2.797373e-06	0.999999
5	1.493340e-07	0.999999

Using PCA also provides a clear method of data visualization, Figure 5.21 plots the Taylor scale data using the top two principal components; here, the axes are the two principal components which are themselves a linear combination of original features, and the color of points corresponds to the value of the Taylor scale. A fairly clear color gradient is visible, which shows the viability of finding a lower-dimensional representation of the data to predict the Taylor scale.

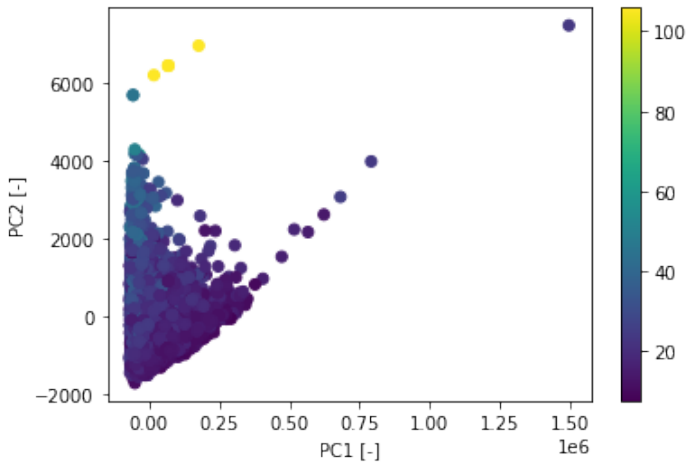


Figure 5.21: Visualization of Taylor scale data using principal components. The first two principal components are used as the axes, providing a lower dimensional visualization that preserves data variance. The color of data points corresponds to the value of the Taylor scale. A clear gradient is visible, showing the possibility of representing the data in a far lower dimension.

Chapter 6

Implementation and results

With the observations from EDA, three machine learning algorithms introduced in chapter 3 are now implemented in order to gain deeper insight into the relationships between the collected features and the calculated Taylor scale.

Models are implemented in Python with `sklearn` and `tensorflow`'s `keras`. A pipeline-based approach is used for tuning hyperparameters for each algorithm, consisting of a standard scaler, then a 5-fold cross-validation. The steps for this are as follows:

- Data is split into a training and test set following a 70-30% split.
- `sklearn StandardScaler` is fit on the training partition and used to transform both the training and test partition. Doing so avoids data leakage from the training set to the test set. Scaling is done according to Equation 6.1.
- The training partition is used in 5-fold cross-validation to obtain average performance measures for possible combinations of hyperparameters.
- The testing partition is used to obtain testing performance on unseen data.

$$z = \frac{x - \mu}{\sigma} \tag{6.1}$$

The setup and results of each machine learning algorithm is first discussed, then the results and findings from all algorithms are compared.

6.1 Lasso regression

A random cross-validation search is done on the following parameters:

- `alpha`: [0.0001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
- `fit_intercept`: [True, False]
- `max_iter` : [1000, 2000, 5000]
- `tol`: [1e-2, 1e-3, 1e-4, 1e-5, 1e-6],
- `positive`: [True, False]

As lasso regression models are fairly fast to train, 500 search iterations are performed to optimize the provided hyperparameters. The optimal set of hyperparameters is as follows:

- `alpha`: 0.01
- `fit_intercept`: True
- `max_iter` : 5000
- `tol`: 1e-4,

- `positive`: False

The result of a 10-fold cross-validation with these hyperparameters are as follows: mean squared error of 13.95 with standard deviation of 2.65, and mean absolute error of 2.68 with standard deviation of 0.09. Figure 6.1 shows a hexagonal binned histogram heatmap of prediction results from the lasso regression model. The red line indicates ideal performance. From the figure, the predictions loosely follow the ideal diagonal, but the density of points is quite uniform over a large area that does not closely follow the line. Also of note is that the large-valued outliers are largely underestimated by the model.

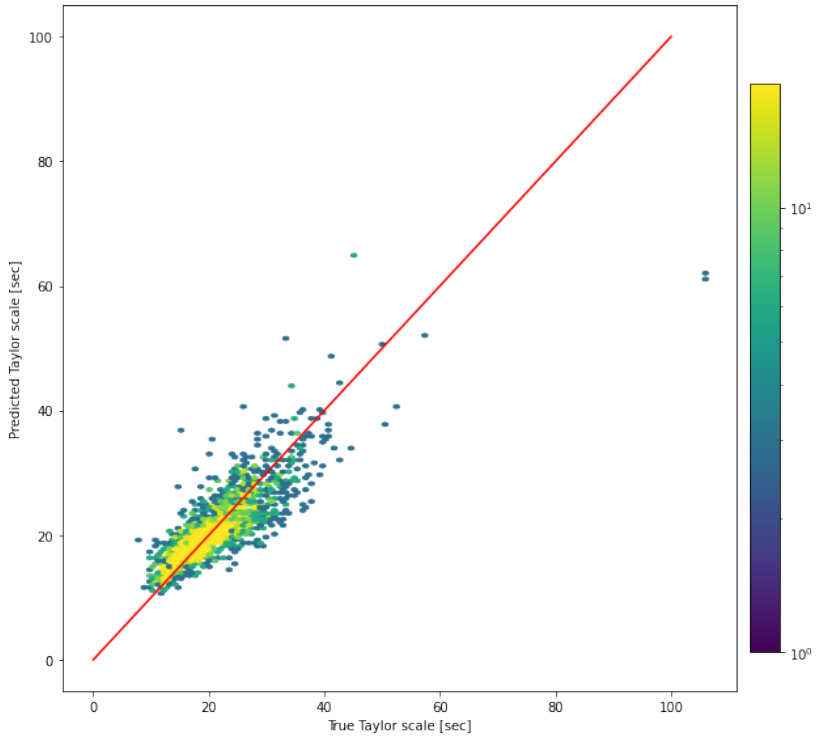


Figure 6.1: Hexagonal binned histogram heatmap of lasso regression prediction performance. The red diagonal represents ideal performance.

The value of a lasso regression is that, as stated in chapter 3, that the regularization method allows for feature coefficients to shrink to zero; such behavior allows for some insight into what the model finds to be the most important features to consider towards a prediction. Table 6.1 shows the average magnitude of the coeffiecient of each feature across the 10 folds of cross-validation. Also of note is that the model intercept always results in a small, near-zero value. As the absolute values of coefficients are taken to obtain such a ranking, an example of a result from one of the ten folds is shown in Table 6.2, where the sign of the coefficient is also visible.

Table 6.1: A ranking of feature importance based on the average magnitude of lasso regression coefficients in a 10-fold cross validation

Feature	**Average coefficient magnitude**
Correlation_timescale	0.8244
Magnetic_field_rms	0.1835
Correlation_timescale_est	0.1668
Electron_velocity_0	0.1069
Proton_density2	0.1018
Electron_density	0.0488
Electron_velocity_1	0.0387
Electric_field	0.0373
Proton_velocity_0	0.0314
Pressure	0.0177
Electron_temp	0.0109
Proton_velocity_1	0.0100
Flow_speed	0.0062
Temperature	0.0021
Electron_velocity_2	0.0008
Plasma_Beta	5.58e-05
Proton_density	0.0
Proton_temp	0.0
Proton_velocity_2	0.0

Table 6.2: Example of lasso regression values one one of the ten folds, also showing the signs of coefficients

Feature	**Coefficient**
Correlation_timescale	0.8300
Correlation_timescale_est	-0.1762
Electron_density	0.0570
Electron_temp	0.0076
Electron_velocity_0	0.1033
Electron_velocity_1	-0.0400
Electron_velocity_2	-0.0
Proton_density	0.0
Proton_temp	0.0
Proton_velocity_0	0.04211
Proton_velocity_1	0.0112
Proton_velocity_2	-0.0
Proton_density2	0.0894
Temperature	0.0
Flow_speed	-0.0
Plasma_Beta	-0.0
Pressure	-0.0175
Electric_field	0.0318
Magnetic_field_rms	-0.1772

6.2 Random forest regression

A random search is done on the following parameters:

- `n_estimators`: [2, 4, 8, 16, 32, 64, 128, 192, 256, 512]
- `criterion`: ['squared_error', 'absolute_error', 'poisson']
- `max_depth`: [None, 3, 5, 7],
- `max_features`: ['auto', 'sqrt', 'log2']

Of particular interest is the claim made by Oshiro et al. [38] regarding diminishing performance and vastly increased computation time when increasing the number of trees in a random forest; though this may seem intuitive, this behavior can be quickly evaluated for this context. Oshiro et al. [38] suggests using between 64 and 128 trees, and that anything beyond would incur unneeded computation time for negligible performance increases. Table 6.3 shows the tabulated results of such an investigation with averaged measurements from 5-fold cross-validation. The results follow the expectation: the mean training time per fold increases quite drastically, while performance plateaus. A bar plot is shown in Figure 6.2 with a finer resolution on lower tree counts. Here, the negative mean absolute error is used as a score metric, but is inverted to support visualization (lower is better). The same effect is visible, successively increasing the tree count will improve performance up to a point of greatly diminished returns, while computation time drastically increases.

Table 6.3: Investigation of the effect of number of trees within a random forest regressor with counts ranging from 16 to 512. Results are averaged from 5-fold cross-validation. In terms of r2 score, a random forest with 192 ranks the highest.

Number of trees	Mean training time [sec]	Mean r2 score [-]	Rank
16	0.757378	0.770980	10
32	1.496976	0.776927	9
64	2.969998	0.780294	8
128	5.952505	0.781260	7
192	9.038451	0.784179	1
256	11.963994	0.783499	2
320	15.098454	0.783093	4
384	17.881315	0.782876	5
448	20.745661	0.782345	6
512	23.574355	0.783215	3

With this result in mind, a random cross-validation search is performed for 200 iterations to optimize the provided hyperparameters. The optimal set of hyperparameters is as follows:

- `n_estimators`: 128
- `criterion`: 'squared_error'
- `max_depth`: None, i.e. unrestrained
- `max_features`: 'auto'

The result of a 10-fold cross-validation with these hyperparameters are as follows: mean squared error of 7.70 with standard deviation of 0.66, and mean absolute error of 1.91 with standard deviation of 0.09. A hexagonal binned heatmap of predictions from the random forest is shown in Figure 6.3. Here, the red diagonal represents cases where the predicted value perfectly matches the true value. The predictions follow the ideal diagonal fairly closely, with much of the density occurring near the line then decreasing outwards.

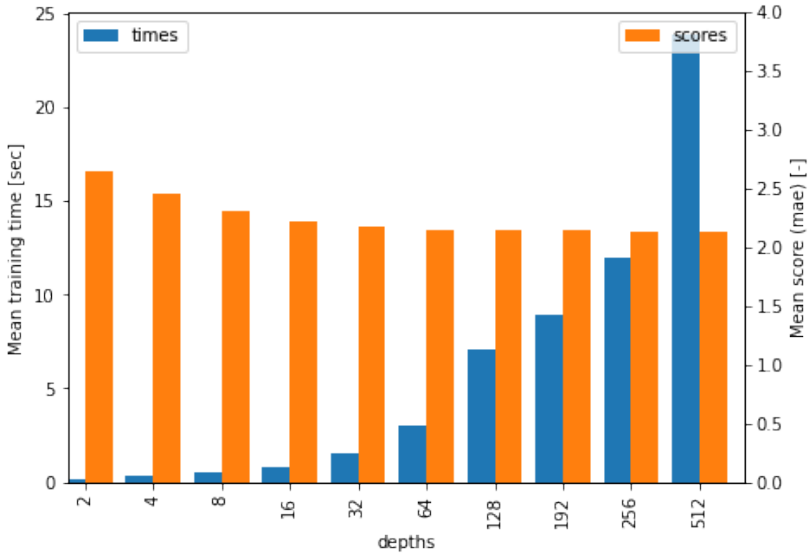


Figure 6.2: Bar plot of negative mean absolute error (inverted for visualization purposes) and computation time, per fold in 5-fold cross-validation. As tree counts increase, a small increase in performance is observed until a point of diminishing return, while runtimes increase dramatically.

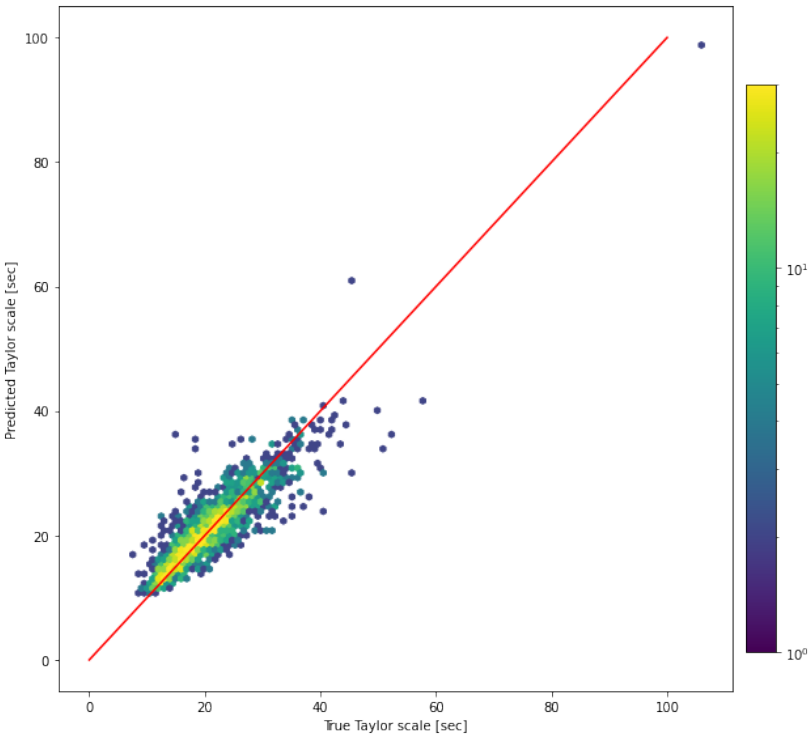


Figure 6.3: Hexagonal binned histogram heatmap of random forest predictions. The red diagonal represents the ideal cases where the predicted value perfectly matches the true value. The heatmap color denotes point density.

To gain more insight into the inner workings of the random forest and the relationships between features to the target variable, an interpretability library, SHAP [46], can be used.

Decision tree structures are inherently interpretable to humans, but having many in an ensemble model becomes more difficult to interpret. Libraries like SHAP allow for interpretation for such complex models. A summary plot from SHAP is shown in Figure 6.4; the SHAP value is a custom importance metric used by SHAP [46]. The plot can be interpreted through both the SHAP value and feature value color bar. Additionally, the features are listed in order of overall importance. As an example, the correlation scale is deemed to be very important to the model prediction; the plot shows that high feature values have a large, positive effect on the model output and vice versa. In other words, a higher correlation scale contributes to a higher Taylor scale prediction and a lower correlation scale contributes to a lower Taylor scale prediction, which is supported by finding from EDA in chapter 5. The most important features according to Figure 6.4 also generally follow findings from chapter 5.

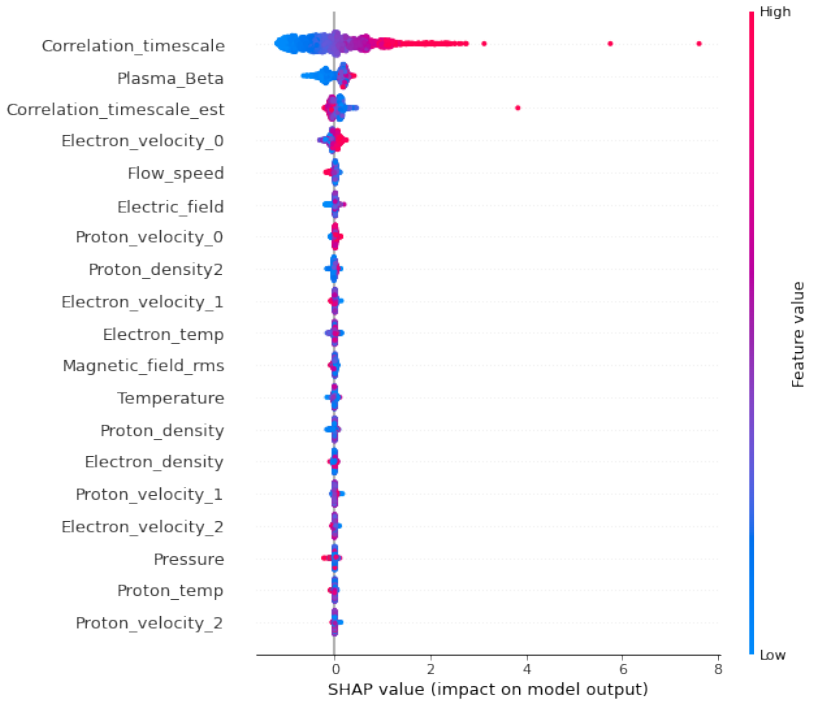


Figure 6.4: SHAP summary plot of the random forest. Features are listed in order of importance, and contributions can be interpreted with the combination of feature values and SHAP values.

6.3 Multi-layered perceptron

Due to the large number of (hyper)parameters to consider when designing an MLP, a grid or random parameter search is too costly to perform. Instead, an architecture is pre-specified, and parameters are selected through some trial-and-error and through comparison with literature. The chosen structure is used with `sklearn`'s `MLPRegressor`; The structure of the MLP is as follows:

- Adam optimizer, learning rate 0.0001
- Glorot uniform kernel initializer [47]
- Fully-connected layer (64), ReLU activation, dropout (0.3)
- Fully-connected layer (512), ReLU activation, dropout (0.3)
- Fully-connected layer (256), ReLU activation, dropout (0.3)

- Fully-connected layer (64), ReLU activation, dropout (0.3)
- Output layer (1), linear activation

Forward feature selection

Before performance is evaluated, a feature selection process is performed to gain some insight into the process behind predicting the Taylor scale through the MLP. This is done with `mlxtend`'s `SequentialFeatureSelector` [48] to perform a forward search.

A forward search begins with an empty set of features, then sequentially iterates through the full set of features, evaluating the increase in improvement resulting from adding each feature individually. The feature that results in the highest increase in performance is then added to the current solution, and the process repeats until the pre-specified number of features is reached. A backwards search follows a similar algorithm, but instead begins with the full set of features then sequentially removes features until the number of features is reached. Naturally, a forward search is more efficient when the speculated number of informative features desired is low compared to the size of the full set, and a backwards search is more efficient when a large number of informative features is expected. Given the findings from the previous lasso regression and random forest approaches indicate that a small subset of features contributes the majority towards the prediction, a forward search is used as opposed to a backwards search.

The forward feature selection algorithm is detailed in Algorithm 1. Here, the solution is initialized as an empty set, $\{\}$, and the iteration count k as zero. Until the number of desired features (equivalent to iteration number) is reached, the performance from the addition of each remaining feature to the current solution is evaluated by $C(\cdot)$; this function is chosen to the the negative mean squared error. The optimal choice of feature to add is that which corresponds to the maximum $C(\cdot)$ score, which is then added to the current solution set. The iteration count k is then incremented. The output is then $Y_d = \{y_1, y_2, ..., y_d\}$, a set of the most important features of size d. Important to note is that both forward and backwards searches are greedy algorithms, meaning that the best option is taken at each iteration without considering previous or potential future steps. In rare cases, an uninformative feature can become informative when combined with another uninformative feature; this can be overlooked by a greedy algorithm. Feature interactions are considered to an extent, however; once a feature is added to evaluate its effect, it must provide information in addition to that provided by the previously chosen features.

Algorithm 1 Forward feature selection

Require: F, the full set of n features $\{f_1, f_2, ..., f_n\}$; d, desired number of features

Initialize $Y_0 = , k = 0$
repeat
  for each $f_j \in F - Y_k$ **do**
    $C_j \leftarrow C(Y_k + f_j)$
  end for
  $f_{opt} \leftarrow argmax(C_j)$
  $Y_{k+1} = Y_k + f_{opt}$
  $k \leftarrow k + 1$
until $k = d$

Applying the forward feature selection to the MLP model specified is a form of wrapper feature selection as a machine learning model is used to evaluate the score of candidate solutions, as opposed to filter feature selection methods which use an alternate metric; as a

result, the selected set of features will be slightly biased towards the type of model used in the evaluation process. By observing the results from previous models, a set of size 5 is used for feature selection; this decision is somewhat arbitrary, but from plots like Figure 6.4 and Table 6.1, a set of five features seems suitable to aim for. The set of selected features in order of their addition is as follows:

- Correlation scale (fit method)
- Correlation scale (1/e method)
- Electron density
- Flow speed
- Plasma Beta

A new data set is then created, keeping only these features, to be used to evaluate the performance of the MLP through 10-fold cross-validation. The MLP is now implemented in `keras` [49] due to the the greater control over network design available. Training is run for 100 epochs, but an early stopper is also implemented; this monitors validation set perfromance during training as a way to gauge the onset of overfitting. Overfitting is generally characterized by a rise in validation loss alongside continued decrease in training loss. The early stopper is set to monitor the mean squared error loss and stops training once validation loss improvement falls below a threshold of 0.01 with a patience of 10 epochs. An example of training performance of one of the folds is shown in Figure 6.5, noting that the loss metric is mean squared error on the standardized data, hence the smaller values. Here, the early stopper terminated training after 25 epochs due to diminished improvements. Also note that validation loss is consistently lower than the training loss, indicating that the model avoids overfitting to the training set; this is due to the inclusion of dropout layers.

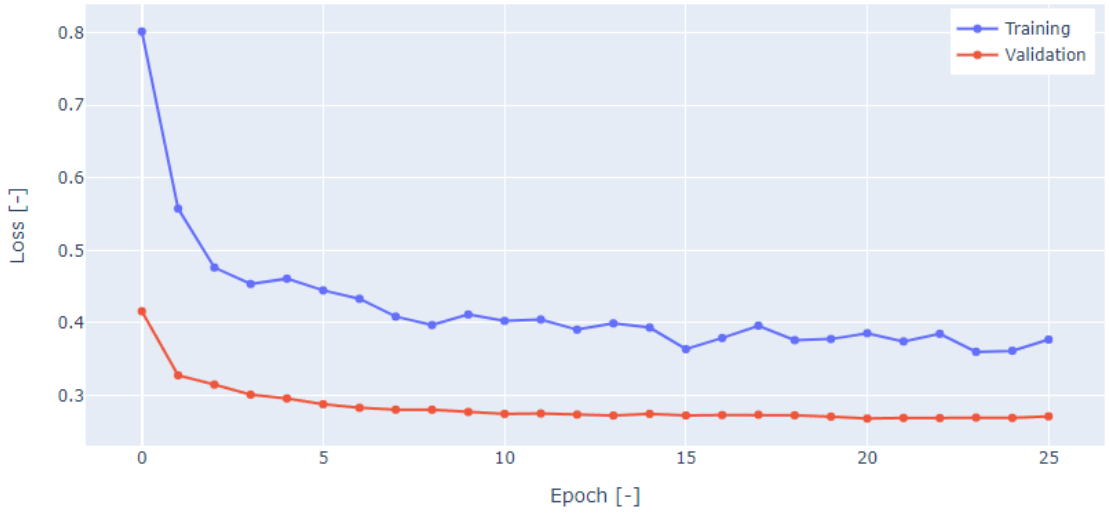


Figure 6.5: Training performance over training epochs. Note that the loss is measured in mean squared error, calculated on normalized data during training; the values are therefore smaller than those reported on unstandardized data.

The performance is as follows: mean squared error of 13.73 with standard deviation 2.81, and mean absolute error of 2.61 with standard deviation of 0.10. Predictive performance is shown in Figure 6.6, where the red diagonal denotes ideal performance. The MLP's predictions follow the diagonal fairly closely, but have a fairly flat density distribution.

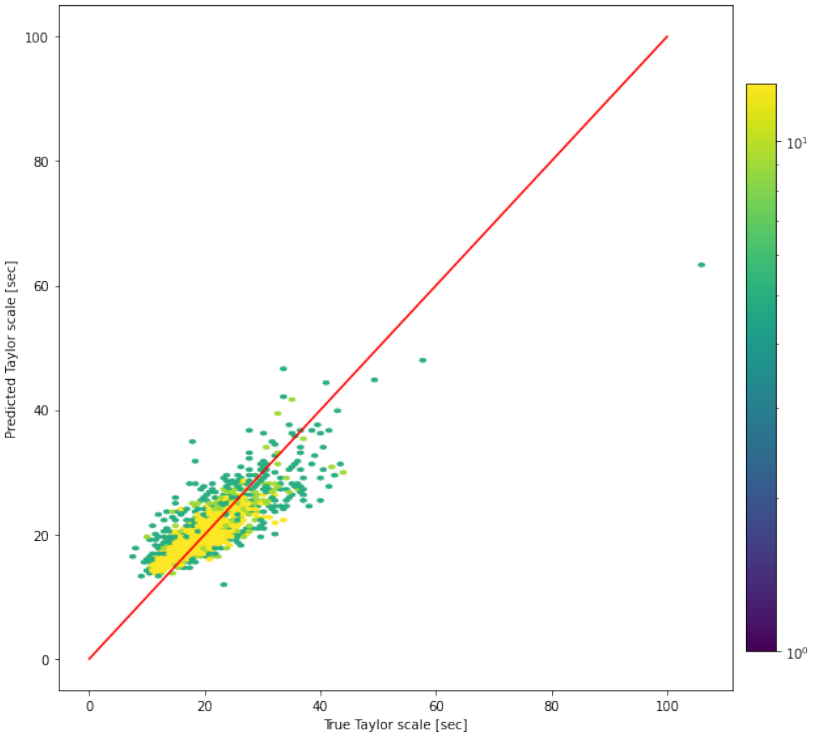


Figure 6.6: Hexagonal binned histogram heatmap of MLP prediction performance. The red diagonal represents the ideal cases where the predicted value perfectly matches the true value. The heatmap color denotes point density.

6.4 Discussion

The results of the three models are compiled in Table 6.4 and Table 6.5. To restate the main goal of this work, machine learning and data mining are used as tools to gain understanding of the relationships of large-scale solar wind attributes and characteristics and the Taylor scale. Parts of this insight was gained through EDA in chapter 5, and here the insight comes in the form of the various interpretability techniques applied to the three machine learning models. The discussion is therefore not explicitly about the performance of each model, but rather what can be derived about the inner workings of the models to arrive at the prediction. As such, Table 6.4 shows performance metrics of each model during 10-fold cross-validation, and Table 6.5 shows the top five ranked features by importance, according to each model.

In terms of complexity and computation time, lasso regression models are the least complex and fastest to train, followed by random forest, then the MLPs. From Table 6.4, the lasso regression model performs the worst of the three with the highest average error metrics. This is likely due to the restrictive nature of its generated model representation; the form is restricted to a linear model with an intercept. Conversely, random forests and MLPs can model non-linear interactions. The benefit of using a lasso regression model is its inherent explainability through the ability to shrink feature coefficients to zero; these indicate weaker, less important features. Next, the MLP is the most complex model of the three and requires the highest computation times. Despite this however, performance is only slightly improved over the lasso regression model, and with a larger deviation of results. This may be due to the comparatively reduced level of hyperparameter optimization due to the high numbers of degrees of freedom and high training times, or due to the often over-confident

nature of neural network predictions [50]. Additionally, MLPs have long been considered black box approaches, though methods have been developed to gain insight into predictions [46] [51]. Finally, the random forest performs the best by a significant margin, with lower deviation in scores. This improved performance over the other models is clear in Figure 6.3, where point density is much more concentrated along the diagonal.

Table 6.4: Comparison of performance metrics of the three models: lasso regression, random forest, and multi-layered perceptron through 10-fold cross-validation

Model	**Mean MSE**	**MSE standard deviation**	**Mean MAE**	**MAE standard deviation**
Lasso regression	13.95	2.65	2.68	0.09
Random forest	7.70	0.66	1.91	0.09
Multi-layered perceptron	13.73	2.81	2.61	0.10

Finally, the results of the various methods of feature importance rankings, shown in Table 6.5, can be compared. The lasso regression model is inherently interpretable through the form of its final model representation, specifically the direction and magnitude of model coefficients. Though individual decision trees are also inherently interpretable, when many are used in an ensemble random forest, this interpretability breaks down. To aid in this, a Python library focused on interpretability, SHAP [46], is used. SHAP provides both feature rankings and insight into how much and in what direction the values of these features contribute to the end prediction. This is seen in Figure 6.4. Finally, MLPs have limited interpretability. Though methods like SHAP [46] and LIME [51] provide methods aid in this, a feature selection approach is instead taken. An MLP is used within a wrapper feature selection process, in which a forward search is done to build a feature set that optimizes performance on the MLP. As mentioned previously, a limitation of all wrapper feature selection methods, which use the model in its candidate evaluation process, the resulting feature set is likely biased towards the type of model used. Still, this provides interesting insight when compared with the results of the other feature rankings.

An immediate observation from Table 6.5 is that all models rank the correlation scale, computed through the fit method, to be the most important feature. This is consistent with finding during EDA. Similarly, all three rankings involve the correlation scale computed through the 1/e method at high rankings. Intuitively, if both methods provide an estimate of the same quantity, it is expected that only one of the two would be chosen; which during EDA it was evident that both quantities had a correlation with the target variable, the inclusion of both could point to the comparatively high correlation compared to less important features despite sharing information with each other. One method was not chosen over the other due to the exploratory nature of this work, so deeper analysis of the suitability of both methods is an area for future work. Next, the electron velocity (first component), plasma beta, and flow speed all appear in two of the three rankings. The inclusion of these is supported by findings in EDA, especially in the correlation matrix in Figure 5.16; such a matrix only considers pairwise interactions however, and not the interaction between feature sets towards the final prediction, which is where these feature ranking methods provide insight into. Finally, the RMS magnetic field, proton density, and electron density appear in one of the three rankings. These features showed non-negligible correlations in the correlation matrix in Figure 5.16, but not as high as some other features. This is again likely due to feature interaction within the feature set, which is not captured within a correlation matrix. Most notably, the RMS magnetic field strength showed a low correlation during EDA, but is

included in the ranking from the lasso regression. Given that the Taylor scale is computed through magnetic field data, the inclusion of this in the ranking is somewhat expected despite the low pairwise correlation. The main findings from this table are that an apparent large connection is observed between the large-scale correlation scale and the small-scale Taylor scale within solar wind, and that some connections can be observed between general solar wind characteristics and this small-scale behavior.

Table 6.5: Comparison of feature importance rankings from each machine learning model: lasso regression, random forest, and multi-layered perceptron through various interpretation methods

Feature rank	**Lasso regression**	**Random forest**	**Multi-layered perceptron**
1	Correlation scale (fit method)	Correlation scale (fit method)	Correlation scale (fit method)
2	Magnetic field RMS	Plasma Beta	Correlation scale (1/e method)
3	Correlation scale (1/e method)	Correlation scale (1/e method)	Electron density
4	Electron velocity (1st GSE component)	Electron velocity (1st GSE component)	Flow speed
5	Proton density	Flow speed	Plasma Beta

Chapter 7

Conclusion

The focus of this work was to explore the nature of turbulent microscales through the application of data mining and machine learning techniques. Specifically, a data set was created through consolidating data from multiple sources focused on measuring solar wind quantities including NASA's Wind spacecraft and the open OMNI data, and computing two turbulent scales through methods used in literature: the correlation scale and Taylor microscale. The purpose of this data set is to investigate the relationship between large scale turbulence characteristics and the Taylor microscale; as such, large-scale averages of solar wind properties are taken over 6-hour intervals over 5 years, with the corresponding Taylor scale calculated over the same intervals. Some exploratory data analysis is then performed on the data, after which machine learning algorithms are applied.

The results come in the form of both observations made during exploratory data analysis and through applying AI explainability techniques to machine learning models. Machine learning models considered include lasso regression, random forests, and multi-layered perceptrons. The predictive power of these trained models in themselves are not the focus, but rather what can be discovered about the relationship between the input features and the Taylor scale through analyzing the inner workings of the trained models. As such, different methods of analysis are used for each model: lasso regression models are inherently interpretable through its trained model coefficients, a Python library SHAP [46] is used to gain insight into the random forest models, and a forward feature selection approach is used for the multi-layered perceptron. These methods result in a ranking of features that contribute the most towards the prediction. Using such methods also provides more insight than pairwise correlation metrics as feature interactions are also taken into account.

The results reveal an apparent strong correlation between the correlation time scale and the Taylor time scale and is ranked as the most important contribution towards the prediction by all three models. Among the other highly ranked features are the electron velocity, plasma beta, and proton and electron densities.

Aspects open to future work exist in both the field of physics and machine learning. Deeper domain knowledge is required to further interpret the implications of findings in this work and links to current theory, especially in the apparent high correlation between the correlation scale and Taylor scale and the most optimal way to compute these scales from the methods considered in this work. Extensions on the machine learning used in this work are numerous. At a basic level, more computation resources can be used to process a larger data set, possibly with more features, and further optimize the used machine learning algorithms. Alternate data set formats can also be considered; instead of single data entries for 6-hour intervals, a more complex representation can be explored. An example lies in using a down-sampled time series of each parameter within the 6-hour interval corresponding to the target Taylor scale; the trend of the features within the period can then be

considered, for example using a convolutional neural network, instead of a single aggregate value. Alternatively, the interval can be analyzed in the frequency domain, resulting in a multi-dimensional spectral input to predict the Taylor scale. The discoveries made in this work show promise for such further investigation on turbulent microscales using machine learning and data mining tools.

Bibliography

[1] S. B. Pope, *Turbulent Flows*. Cambridge University Press, Aug. 2000.

[2] R. Bruno and V. Carbone, "The solar wind as a turbulence laboratory," *Living Reviews in Solar Physics*, vol. 10, 2013.

[3] R. Bandyopadhyay, W. H. Matthaeus, A. Chasapis, C. T. Russell, R. J. Strangeway, R. B. Torbert, B. L. Giles, D. J. Gershman, C. J. Pollock, and J. L. Burch, "Direct measurement of the solar-wind taylor microscale using MMS turbulence campaign data," *The Astrophysical Journal*, vol. 899, p. 63, Aug. 2020.

[4] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] J. Amaya, R. Dupuis, M. E. Innocenti, and G. Lapenta, "Visualizing and interpreting unsupervised solar wind classifications," *Frontiers in Astronomy and Space Sciences*, vol. 7, Sep 2020.

[6] H. Breuillard, R. Dupuis, A. Retino, O. Le Contel, J. Amaya, and G. Lapenta, "Automatic classification of plasma regions in near-earth space with supervised machine learning: Application to magnetospheric multi scale 2016–2019 observations," *Frontiers in Astronomy and Space Sciences*, vol. 7, 2020.

[7] R. L. Bailey, M. A. Reiss, C. N. Arge, C. Möstl, C. J. Henney, M. J. Owens, U. V. Amerstorfer, T. Amerstorfer, A. J. Weiss, and J. Hinterreiter, "Using gradient boosting regression to improve ambient solar wind model predictions," *Space Weather*, vol. 19, May 2021.

[8] K. Kawamura, T. Nishigaki, A. Andriamananjara, H. Rakotonindrina, Y. Tsujimoto, N. Moritsuka, M. Rabenarivo, and T. Razafimbelo, "Using a one-dimensional convolutional neural network on visible and near-infrared spectroscopy to improve soil phosphorus prediction in madagascar," *Remote Sensing*, vol. 13, p. 1519, Apr. 2021.

[9] K. Wang, P. Guo, and A.-L. Luo, "A new automated spectral feature extraction method and its application in spectral classification and defective spectra recovery," *Monthly Notices of the Royal Astronomical Society*, vol. 465, pp. 4311–4324, Dec. 2016.

[10] B. Jiang, X. Fang, Y. Liu, X. Wang, and J. Liu, "Spectral feature extraction using partial and general method," *Advances in Astronomy*, vol. 2021, pp. 1–10, June 2021.

[11] J. Deng, W. Song, D. Liu, Q. Li, G. Lin, and H. Wang, "Improving the spatial resolution of solar images using generative adversarial network and self-attention mechanism," *The Astrophysical Journal*, vol. 923, p. 76, Dec. 2021.

[12] "What is a lagrange point?." https://solarsystem.nasa.gov/resources/754/what-is-a-lagrange-point/.

[13] L. Franci, S. Landi, L. Matteini, A. Verdini, and P. Hellinger, "PLASMA BETA DEPENDENCE OF THE ION-SCALE SPECTRAL BREAK OF SOLAR WIND TURBULENCE: HIGH-RESOLUTION 2d HYBRID SIMULATIONS," *The Astrophysical Journal*, vol. 833, p. 91, dec 2016.

[14] E. Marsch, "Kinetic physics of the solar corona and solar wind," *Living Reviews in Solar Physics*, vol. 3, 2006.

[15] H. Tennekes and J. L. Lumley, *A First Course in Turbulence*. The MIT Press, 03 1972.

[16] W. H. Matthaeus, J. M. Weygand, P. Chuychai, S. Dasso, C. W. Smith, and M. G. Kivelson, "Interplanetary magnetic taylor microscale and implications for plasma dissipation," *The Astrophysical Journal*, vol. 678, pp. L141–L144, Apr. 2008.

[17] H. Tennekes and J. L. Lumley, *A First Course in Turbulence*. The MIT Press, 1972.

[18] R. Bandyopadhyay, M. L. Goldstein, B. A. Maruca, W. H. Matthaeus, T. N. Parashar, D. Ruffolo, R. Chhiber, A. Usmanov, A. Chasapis, R. Qudsi, S. D. Bale, J. W. Bonnell, T. D. de Wit, K. Goetz, P. R. Harvey, R. J. MacDowall, D. M. Malaspina, M. Pulupa, J. C. Kasper, K. E. Korreck, A. W. Case, M. Stevens, P. Whittlesey, D. Larson, R. Livi, K. G. Klein, M. Velli, and N. Raouafi, "Enhanced energy transfer rate in solar wind turbulence observed near the sun from parker solar probe," *The Astrophysical Journal Supplement Series*, vol. 246, p. 48, Feb. 2020.

[19] T. N. Parashar, M. Cuesta, and W. H. Matthaeus, "Reynolds number and intermittency in the expanding solar wind: Predictions based on voyager observations," *The Astrophysical Journal*, vol. 884, p. L57, Oct. 2019.

[20] R. Chhiber, M. L. Goldstein, B. A. Maruca, A. Chasapis, W. H. Matthaeus, D. Ruffolo, R. Bandyopadhyay, T. N. Parashar, R. Qudsi, T. D. de Wit, S. D. Bale, J. W. Bonnell, K. Goetz, P. R. Harvey, R. J. MacDowall, D. Malaspina, M. Pulupa, J. C. Kasper, K. E. Korreck, A. W. Case, M. Stevens, P. Whittlesey, D. Larson, R. Livi, M. Velli, and N. Raouafi, "Clustering of intermittent magnetic and flow structures near parker solar probe's first perihelion—a partial-variance-of-increments analysis," *The Astrophysical Journal Supplement Series*, vol. 246, p. 31, Feb. 2020.

[21] T. N. Parashar, M. L. Goldstein, B. A. Maruca, W. H. Matthaeus, D. Ruffolo, R. Bandyopadhyay, R. Chhiber, A. Chasapis, R. Qudsi, D. Vech, D. A. Roberts, S. D. Bale, J. W. Bonnell, T. D. de Wit, K. Goetz, P. R. Harvey, R. J. MacDowall, D. Malaspina, M. Pulupa, J. C. Kasper, K. E. Korreck, A. W. Case, M. Stevens, P. Whittlesey, D. Larson, R. Livi, M. Velli, and N. Raouafi, "Measures of scale-dependent alfvénicity in the first PSP solar encounter," *The Astrophysical Journal Supplement Series*, vol. 246, p. 58, Feb. 2020.

[22] N. M. BALL and R. J. BRUNNER, "DATA MINING AND MACHINE LEARNING IN ASTRONOMY," *International Journal of Modern Physics D*, vol. 19, pp. 1049–1106, July 2010.

[23] S. Pandey, J. Schumacher, and K. R. Sreenivasan, "A perspective on machine learning in turbulent flows," *Journal of Turbulence*, vol. 21, pp. 567–584, Apr. 2020.

[24] J. H. King, "Solar wind spatial scales in and comparisons of hourly wind and ACE plasma and magnetic field data," *Journal of Geophysical Research*, vol. 110, no. A2, 2005.

[25] N. Papitashvili and J. King, "A draft high resolution omni data set," *AGU Spring Meeting Abstracts*, 05 2006.

[26] M. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, Association for Computational Linguistics, 2016.

[27] L. Gallana, F. Fraternale, M. Iovieno, S. M. Fosson, E. Magli, M. Opher, J. D. Richardson, and D. Tordella, "Voyager 2 solar plasma and magnetic field spectral analysis for intermediate data sparsity," *Journal of Geophysical Research: Space Physics*, vol. 121, pp. 3905–3919, May 2016.

[28] G. Huang, "Missing data filling method based on linear interpolation and lightgbm," *Journal of Physics: Conference Series*, vol. 1754, p. 012187, Feb. 2021.

[29] K. Zor, O. Çelik, O. Timur, B. Yıldırım, and A. Teke, "Simple approaches to missing data for energy forecasting applications," 05 2018.

[30] Y. NARIYUKI, M. SASAKI, T. HADA, and S. INAGAKI, "Reconstruction of time series observed in linear magnetized plasma PANTA via a machine learning algorithm," *Plasma and Fusion Research*, vol. 14, pp. 1301157–1301157, Oct. 2019.

[31] A. Dash, J. Ye, and G. Wang, "High resolution solar image generation using generative adversarial networks," 2021.

[32] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[33] M. Hapgood, "Space physics coordinate transformations: A user guide," *Planetary and Space Science*, vol. 40, pp. 711–717, May 1992.

[34] "Spdf about omniweb data." https://omniweb.gsfc.nasa.gov/html/omni_min_data.html.

[35] G. A. Gary *Solar Physics*, vol. 203, no. 1, pp. 71–86, 2001.

[36] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[37] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, pp. 197–227, Apr. 2016.

[38] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?," in *Machine Learning and Data Mining in Pattern Recognition*, pp. 154–168, Springer Berlin Heidelberg, 2012.

[39] L. B. Wilson, A. L. Brosius, N. Gopalswamy, T. Nieves-Chinchilla, A. Szabo, K. Hurley, T. Phan, J. C. Kasper, N. Lugaz, I. G. Richardson, C. H. K. Chen, D. Verscharen, R. T. Wicks, and J. M. TenBarge, "A quarter century of wind spacecraft discoveries," *Reviews of Geophysics*, vol. 59, May 2021.

[40] A. Koval and A. Szabo, "Magnetic field turbulence spectra observed by the wind spacecraft," in *AIP Conference Proceedings*, AIP, 2013.

[41] L. Vuorinen, H. Hietala, and F. Plaschke, "Jets in the magnetosheath: IMF control of where they occur," *Annales Geophysicae*, vol. 37, pp. 689–697, Aug. 2019.

[42] M. V. Vokhmyanin, N. A. Stepanov, and V. A. Sergeev, "On the evaluation of data quality in the OMNI interplanetary magnetic field database," *Space Weather*, vol. 17, pp. 476–486, Mar. 2019.

[43] M. Ashour-Abdalla, R. J. Walker, V. Peroomian, and M. El-Alaoui, "On the importance of accurate solar wind measurements for studying magnetospheric dynamics," *Journal of Geophysical Research: Space Physics*, vol. 113, pp. n/a–n/a, Aug. 2008.

[44] "Spdf - coordinated data analysis web (cdaweb)." http://cdaweb.gsfc.nasa.gov/.

[45] V. Cox, "Exploratory data analysis," in *Translating Statistics to Make Decisions*, pp. 47–74, Apress, 2017.

[46] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[47] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[48] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, Apr. 2018.

[49] F. Chollet *et al.*, "Keras," 2015.

[50] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, pp. 828–841, Oct. 2019.

[51] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016.