POLITECNICO DI TORINO

Master Degree Course in Computer Engineering

Master Degree Thesis

Deep Neural Networks for Prediction of Geomagnetic Events from Solar Wind Data

Supervisor
Prof. Enrico Magli

Candidate
Chiara Di Nardo

December 2018

Abstract

In the last years a branch of physics, called Heliophysics, concentrated its work in studying the Sun due to the considerable importance this star has for us. Many missions started and spacecrafts were sent to gather as much information as possible to try to have an insight on the phenomena occurring on its surface. Due to the increase of satellites and aerospatial traffic, nowadays the researches don't have the sole purpose to comprehend the star, but also to forecast its events to protect the instruments and humans in space. The thesis' work is located in this ambit, its prime goal is to use an approach based on the latest findings of the deep learning community to classify significant episodes involving the Earth's magnetic field.

The Disturbance Storm-Time index (DST) is a useful hint about the way the magnetic field is affected by the solar wind and the phenomena occurring on the solar corona, such as the coronal mass ejections (CME), and for this reason we used it as label for our classification problem. Mainly the DST is the summary of the metrics collected by near equatorial observatories reflecting the influence the solar wind has on out planet. Thus, the solar wind parameters, collected by the spacecrafts in situ, are used as features composing the datasets given as input to the neural networks. The solar wind parameters are a composition of its proton parameters and magnetic field, each of them with the respective vector and modular components.

The work produced in this thesis is the combination of different phases, one built on top of the other to chose the best way to represent the starting dataset given to us and the best neural network architecture among the suitable ones, to have good prediction results. The phases of our work can be synthetised in the following ones

- Statistical analysis on the initial dataset to understand how is composed and which are its characteristics.
- Application of data augmentation techniques to prepare different datasets to highlight different characteristics of the original one
- Choice of the best dataset among the ones prepared using a simple neural network

- Choice of the best neural network architecture among the ones proposed for our purpose, using the dataset chose at the preceding step evaluated considering different available metrics
- Selection of how much in the future the forecasting is possible, analysing not only the outcomes from the neural networks but also the statistical analysis conducted in the past steps.

For each of the described phases the outcomes are described as well as the difficulties encountered and how they were addressed.

The results conducted on the datasets showed the predilection towards more complicated ones, with a particular structure, but also the inutility in having a huge window of observations since they do not increase the quality of the classification made by the neural network. This made us lean on datasets with less but more significant samples. On the other hand the usage of a good neural network influences the quality of the classification due to its ability in recognising the pattern in the dataset, but also in considering the entire window given and not only the last samples. When coming to forecasting the amount of time in the future we want to forecast impacts the quality of the predictions computed, especially when using samples really further in time from the interval given as input.

Acknowledgements

At the end of my thesis, I would like to thank my supervisor, Prof. Enrico Magli, and Tomas Bjorklund, for giving me the opportunity to prove myself in such a particular project and for their helpful advises during its development.

I would like to thank Gianalfredo Nicolini and Daniele Telloni, from the Osservatorio Astrofisico di Torino, and ALTEC for their aid in understanding the physical phenomena going on under the simple data we received, giving us an insight on the complexity of such essential and yet unknown events that are part of our daily lives.

Contents

Chapter 1

Introduction

Sun and its related phenomena such as solar flares and coronal mass ejections (CME) can dangerously impact Earth and its near space environment, affecting its atmosphere and also the geomagnetic field that surround it, putting in danger not only technology but also society. Numerous are the researches started along the years to try to understand better the phenomena happening on Sun's surface, which in reality are a consequence of nuclear fusions occurring in its core and their energy spreading through its structure until reaching the outermost part. The events of our interest, solar wind and CME, are generated due to the release of the energy accumulated in the inner part of the star and follow the solar cycle which lasts 11 years.

Every cycle has its own characteristics and scientists work to improve our ability to predict the strength and duration of them, mainly because our ability to forecast this event can help us protect properly radio communications on Earth and also astronauts, for instance. It can also be dangerous to not shielded satellites, electrical transmission line facilities and disrupt radio transmissions, causing radio blackouts, due to the alteration of the ionosphere.

Many are the researches conducted along the years to study this phenomena and try to forecast them applying the deep learning knowledge to such complicated subject, an example are Monica G Bobra and Couvidat, 2015 and M. G. Bobra and Ilonidis, 2016 which are two different studies based on the same dataset, the SDO/HMI vector magnetic field data, but tackling two different sides of the same problem. In the first one a Support Vector Machine (SVM) is used to predict solar flares while in the second one the same machine learning network is used to understand which features are mainly involved in the creation of solar flares and which of them will lead to CMEs. Another study, based on the same data, was conducted by Florios et al., 2018 with the intent of forecasting solar flare comparing different machine learning algorithms to non machine learning ones. The main subject of all these researches is the SDO/HMI vector magnetic field data, also called magnetogram, a pictorial representation of the solar magnetic field, using it

directly or its derived features.

Our work is introduced in this scenario with the aim of carrying a different point of view both in the approach used and also in the dataset implied, comprising features coming from different spacecraft and having as forecasting goal not the occurrence of the phenomena itself, but of its consequences on the Earth. Another aspect of our work is given by the temporal characteristic of our data and the lack of images and visual representations, but the use of only numerical data.

1.1 Contents of the thesis

Chapter 2 describes the Sun as the star of our system, its parts and the inner activities that take part in the creation of the physical phenomena we are interested in, together with a description of how they affect our life and activities on the Eath and a briefly introduction to NASA's missions involved.

In chapter 3 we present the starting, raw, data. Here we study their characteristics, from a statistical point of view, analysing the issues and addressing the problems encountered while during their manipulation to make them suitable and well-formed as input for a neural network. Every issue has been investigated and the available solutions evaluated to have the best outcome.

Chapter 4 introduces all the technologies involved in the development of the thesis and while presenting the common neural networks used in forecasting problems, we present our architecutures, explaining why they were chosen and their possible weakness. We also illustrate which metrics are used to evaluate the results, apart from the standard one, and compare them.

Chapter 5 illustrates the steps taken to chose the best dataset and neural network model between the ones proposed. This process involves the experiments conducted among all the datasets and the neural networks created, based on the comparison of the results evaluated with the chosen metrics. After the selection of the best dataset and neural network we proceeded with the fine tuning of the hyperparameters involved to enhance the forecasting ability.

In chapter 6 we depict the conclusions of our work and describe our proposal for further researches and studies.

Chapter 2

Sun and its activity

The Sun is known to be the star at the centre of the solar system. It influences every aspect of our life, from economics to culture, and its effect reaches even Neptune and Pluto, the furthest planetary body of the solar system. In this chapter, we present the characteristics of the Sun and the solar events occurring on its corona that are of significant interest to the thesis' study.

2.1 The structure of the Sun

The Sun is a yellow dwarf star and is composed mostly by ionised chemical elements like hydrogen and helium and heavier elements, as metals, for less than the 2% of its mass. Its essential elements combine into what is called plasma, which through its convective motion is responsible for the magnetic field characterising the star. The Sun's structure is composed of several layers that actively concur to produce its energy, in the form of photons:

- Core: composing the innermost 20-25% of the radius, it is 150 times denser than water and the place where nuclear fusions take place due to the high temperature, around 22 million degrees Celsius, and pressure. Here hydrogen is transformed into helium through an exothermic reaction.
- Radiative zone: going from the radius' 20-25% to the 70%, in this region the energy coming from the core is slowly transferred by mean of radiation since convection is not possible due to the distance from the surface.
- Tachocline: the boundary between the radiative and convective regions.
- Convective zone: comprehending the region from the 70% of the radius to the point close to the visible surface it transports the energy from the radiative zone to the surface through convection cells.

- Visible portion: since the Sun does not have a defined surface, due to its volatile nature, we can distinguish only the parts visible from us:
  - Photosphere: the deeper of the visible parts composed of granules, columns of hotter gases surrounded by narrow cooler areas. Here the pressure is less than 10% of Earth's pressure at sea level, and the density is ten-thousand of Earth's atmospheric density at sea level. The Sun does not have a solid surface like Earth's one, but this layer is recognised as its surface, in this sense the ray of the star is measured considering it as a limit.
  - Atmosphere: a sort of halo surrounding the Sun, transparent to most visible radiation, composed by the chromosphere, solar transition region, corona and heliosphere. This is the most important layer of the Sun for our work, in particular, the corona is the region of our interest since is the place where the events studied takes place and from there spread in the medium interstellar, impacting with the atmosphere of our planet.

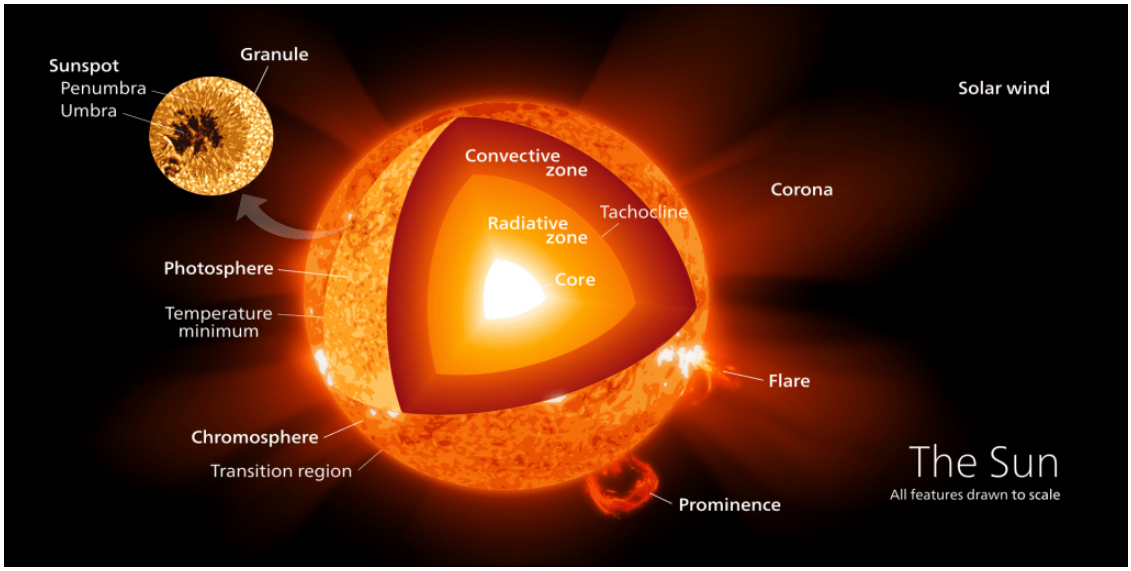


Figure 2.1: Sun structure (Wikipedia Commons/kelvinsong)

2.2 Sun cycle

The solar cycle lasts 11 years and marks the variations in Sun's activity, for example in the levels of solar radiation and material ejection, and also in its appearance, as in the number of sunspots and flares.

The solar cycle can be divided into two parts, recognisable by the number of sunspots (darker spots on the Photosphere) visible on the surface:

- Solar minimum: a period in which for entire months and years there are no sunspots on the photosphere. In this phase, the number of coronal holes, dark regions where the temperature and density are low, and the magnetic field is weak, increase near the poles.
- Solar maximum: the most significant moment of the Sun's cycle, during this period the number of solar flares and coronal mass ejections (CME) increase, sending a considerable amount of solar material into space. The quantity of material sent into space arises because the solar wind's density decreases along with its power to block them.

At the end of each cycle, there is a reorganisation of the Sun's magnetic field, causing the inversion of the poles. Every cycle is different from the other and scientists work to improve our ability to predict the strength and duration of them.

2.3 Solar Wind

The solar wind is a continuous stream of plasma, composed mainly of electrons, protons and alpha particles, varying over the 11-years cycle of Sun's activity in density, temperature and solar latitude and longitude. Thanks to coronal holes, where the usually closed loops of the magnetic field on the Sun's surface, are open into space, the fast component of the solar wind is accelerated spreading coronal matter in the interstellar medium.

2.4 Coronal mass ejection

Sometimes the solar atmosphere enclosed in magnetic loops present on the Sun's surface suddenly and violently release magnetised plasma, composed of charged particles and electromagnetic radiation, into the solar wind. This phenomenon is called coronal mass ejection (CME) and it is widespread during the period of activity of the Sun known as *solar maximum*. These particles flow outward into the solar system at a speed of about 400 kilometres per second.

Three are the features characterising a coronal mass ejection:

- Cavity of low electron density
- Dense core
- Bright leading edge

A CME is often associated with solar flares and is present during a solar prominence, although they are present even during other forms of solar activity, making difficult to define a broad theoretical understanding of it.

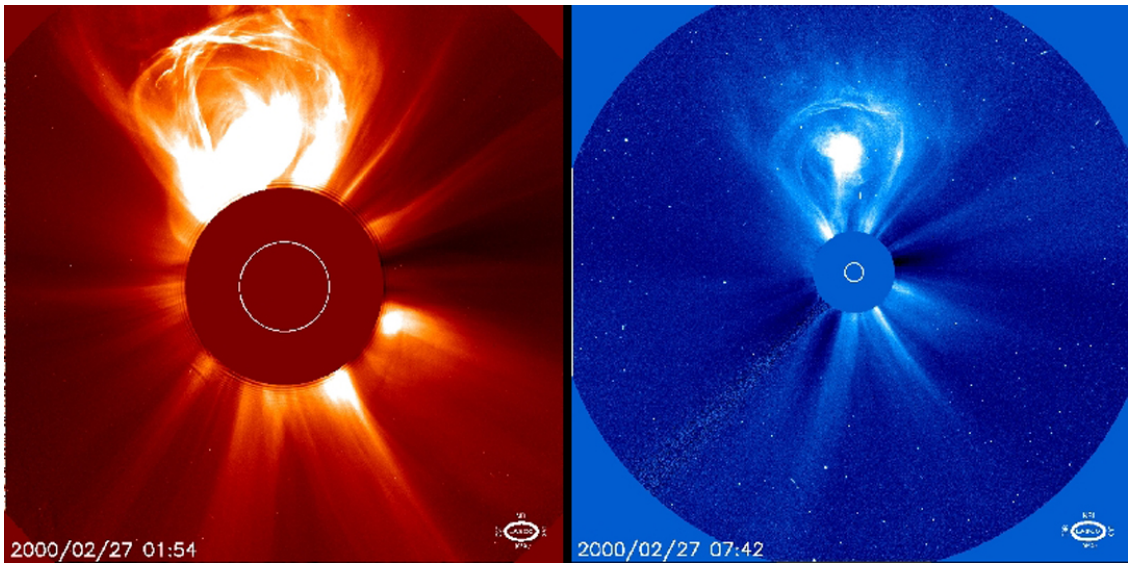


Figure 2.2: Coronal Mass Ejection

Closely linked to CME is the phenomenon called *magnetic reconnection*, the event for which there is a sudden rearrangement of the magnetic field lines when two oppositely directed magnetic fields collide. This reconnection releases the energy contained inside the stressed magnetic fields which become twisted forming a helical structure. Becoming more and more twisted, the CME forms as a way to release the increasing magnetic field and energy, expanding outwards. For this is the reason why coronal mass ejections and solar flares erupt from the active regions on the Sun's surface where magnetic fields are stronger.

2.5 Impact on Earth

Due to the continuous activity of the Sun, the solar wind is always present it the interplanetary atmosphere, reaching the Earth and the other planets of the solar system. The magnetic field blocks the majority of the carried plasma, so humans and all the devices present on the surface of our planet are protected.

Although when CME's shock wave impact with the Earth's magnetic field causes an event called *geomagnetic storm*, a disturbance in the Earth's magnetosphere. From a physic point of view, the Earth's magnetosphere is compressed on the day side and extended on the night side, creating a magnetic tail. When on the night side the particles stream back to the poles, through *magnetic reconnection*, they release energy generating massive visual phenomena like aurorae Australis and Borealis.

Despite the magnificence of this events, it could cause huge damages to humans to higher altitudes as astronauts or people on aeroplanes since the material with

which the means are built can't reduce the unleashed energy, especially when a CME occurs. It can also cause danger to not shielded satellites, electrical transmission line facilities and disrupt radio transmissions, causing radio blackouts, due to the alteration of the ionosphere.

These are the reason why the primary objective of scientists is to study the interactions between the Sun and our planet, potentially forecasting their occurrences to properly shield humans and devices.

2.6 Heliophysics

The study of the Sun and its interaction with the Earth goes under the name of Heliophysics. This branch sees the Sun, the Earth and the planetary environments as an interconnected macro-system, where all its components interact. It studies how the Sun influences the solar system and in particular the Earth itself, providing not only cultural but also economic and political impact in modern society. Sunlight can be thought of as the base of life, but, as said above, it can also produce radiation and magnetic energy which can damage and disrupt a planet's atmosphere and life on it. Even near-Earth the material ejected by the Sun can interfere with our communications, satellites and power grids when too extreme. For these reasons, heliophysics can help us to understand better the nature of this star and, in addition, to protect humans and devices both in space and atmosphere. To this aim, heliophysics exploits all the research subjects that we discussed in the previous sections which are sustained from the collection of related data by the number of spacecraft together with the theoretical knowledge acquired through the years.

2.6.1 Solar observation missions

The presence of solar winds and solar activities is known since the 1950s, but scientist still does not know how they evolve. Therefore, a group of spacecrafts is involved in the Heliophysics flight missions, forming a single observatory called Heliophysics System Observatory (HSO), which enables a vast investigation thanks to their collaboration. The HSO gains its power mostly by being composed of single different missions, which gives it enough distribution, flexibility and the capacity of evolving easily at each addition of a new mission.

Of the many missions involved in the HSO, in the following sections, we discuss in particular of two of them, which gathers the data of our interest: ACE and WIND.

2.6.2 ACE

ACE, standing for Advanced Composition Explorer, observes the particles arriving on the Earth from the Sun and interstellar and galactic sources. It bears six

high-resolution and three monitoring instruments that sample a wide range of energy, both low energy from the Sun and high energy from galactic particles. Being positioned approximately at 1/100 of the distance between Earth and Sun, it has a privileged position and can provide near-real-time information, warning of geomagnetic storms with an advance of about an hour. The main objective of ACE is to measure and compare the composition of different samples of matter, from the solar corona to galactic matter.

The study of the material contributes to our understanding of the formation and evolution of the solar system and the astrophysical process involved. Moreover, it can provide help in shielding astronauts and protect power grids from possible overloading leading to the destruction of communications on Earth.

During the mission period, huge progress has been made, but the cyclic solar activity itself can help make new discoveries and take the research a step further. New challenges are always on the way and one of them is to reach the knowledge neit iseded to forecast space weather fully.

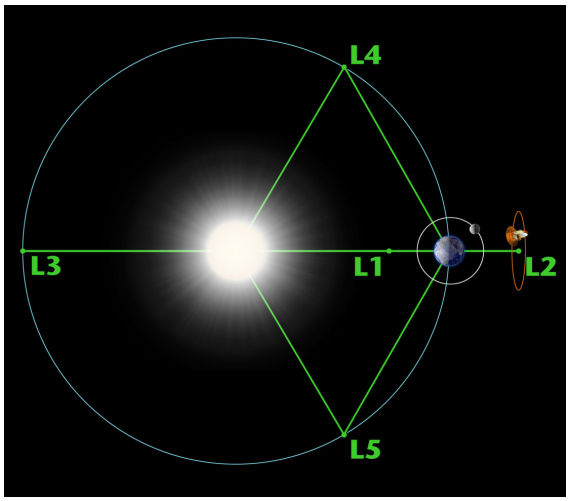


Figure 2.3: Lagrange points

2.6.3 WIND

WIND is one of the first two missions, launched on 1 November 1994, and it is a spacecraft positioned from 2004 in the Earth's L_1 Lagrange point (Lagrangian points, also called oscillation points, are points in space where the gravitational forces of two planetary bodies with a huge mass allow another one, with smaller mass, to maintain a stable position with respect them). It monitors radio waves and plasma in the solar wind about to impact Earth, as an interplanetary component of the International Solar Terrestrial Physics (ISTP) program. WIND carries

instruments to analyse both the magnetic field (Magnetic Field Investigation, MFI) and solar wind (Solar Wind Experiment, SWE) with the objective of gather

- Plasma, energetic particle and magnetic field input for magnetospheric and ionospheric studies.
- Investigate plasma processes occurring in the near-Earth solar wind.
- Determine the magnetospheric output to interplanetary space in the upstream region.

2.6.4 Current algorithms

ALTEC, in collaboration with the Italian Institute of Astrophysics (INAF), is currently directly involved in the development of an algorithm able to predict the events described in this chapter that are strictly correlated with the Disturbance Storm Time (DST) index value. The DST index measures the intensity of the so-called "ring current", an electric current carried by charged particles trapped in a planet's magnetosphere, derived from a network of near-equatorial geomagnetic observatories and capable of shielding the Earth's lower latitudes.

Other studies have been conducted with the same objective, but using different kind of data, such as images taken from chronographs, and not raw data as done in this work.

Chapter 3

Datasets and data augmentation

In this chapter, we describe the data involved in the project, their characteristics and how certain peculiarities have been addressed to make them suitable for the chosen neural network models.

3.1 Dataset composition

The initial data are the raw samples collected from the NASA's spacecrafts in situ and the Disturbance Storm-Time Index (DST) values collected on Earth's observatory. Since the samples follow a temporal order, we interpret the dataset as a time series and all the analysis and methods applied reflect this assumption.

3.1.1 Solar wind experiment data

Solar Wind Experiment (SWE) and magnetic field data are gathered from the spacecrafts and they constitute the features of our dataset, used to train the neural networks. Their components are the following:

- The proton parameters of the solar wind obtained from non-linear fitting to the ion current distribution function (CDF) and with moment techniques. In particular, they are:
  - The three spatial components of the solar wind's speed, expressed in [km/s]:
    * P+_VX_GSE_NONLIN
    * P+_VY_GSE_NONLIN
    * P+_VZ_GSE_NONLIN

  - P+_W_NONLIN: scalar (or isotropic) proton thermal speed [km/s] taken from the trace of anisotropic temperatures

  - Two components of the proton thermal speed [km/s] and its density cm^{-3}:

    * P+_WPERP_NONLIN: Proton thermal speed perpendicular to the magnetic field direction

    * P+_WPAR_NONLIN: Proton thermal speed parallel to the magnetic field direction

    * P+_DENSITY: Proton number density Np from non-linear analysis

- Three components of the magnetic field, averaged over plasma measurement, calculated from the 3-second Magnetic Field Investigation (MFI) experiment:

  - Bx [nT]

  - By [nT]

  - Bz [nT]

  They are expressed from the Geocentric Solar Ecliptic system (GSE) point of view, a reference system based on the Earth-Sun line where the X-axis is towards the Sun and the Z-axis is perpendicular to the plane of the Earth's orbit around the Sun (positive North).

Each parameter is showed in figure 3.1, they are sampled every 92 seconds starting from the 01 January 2005 and for each epoch time the corresponding feature sample. In the figure the holes correspond to the cases in which the data are not present.

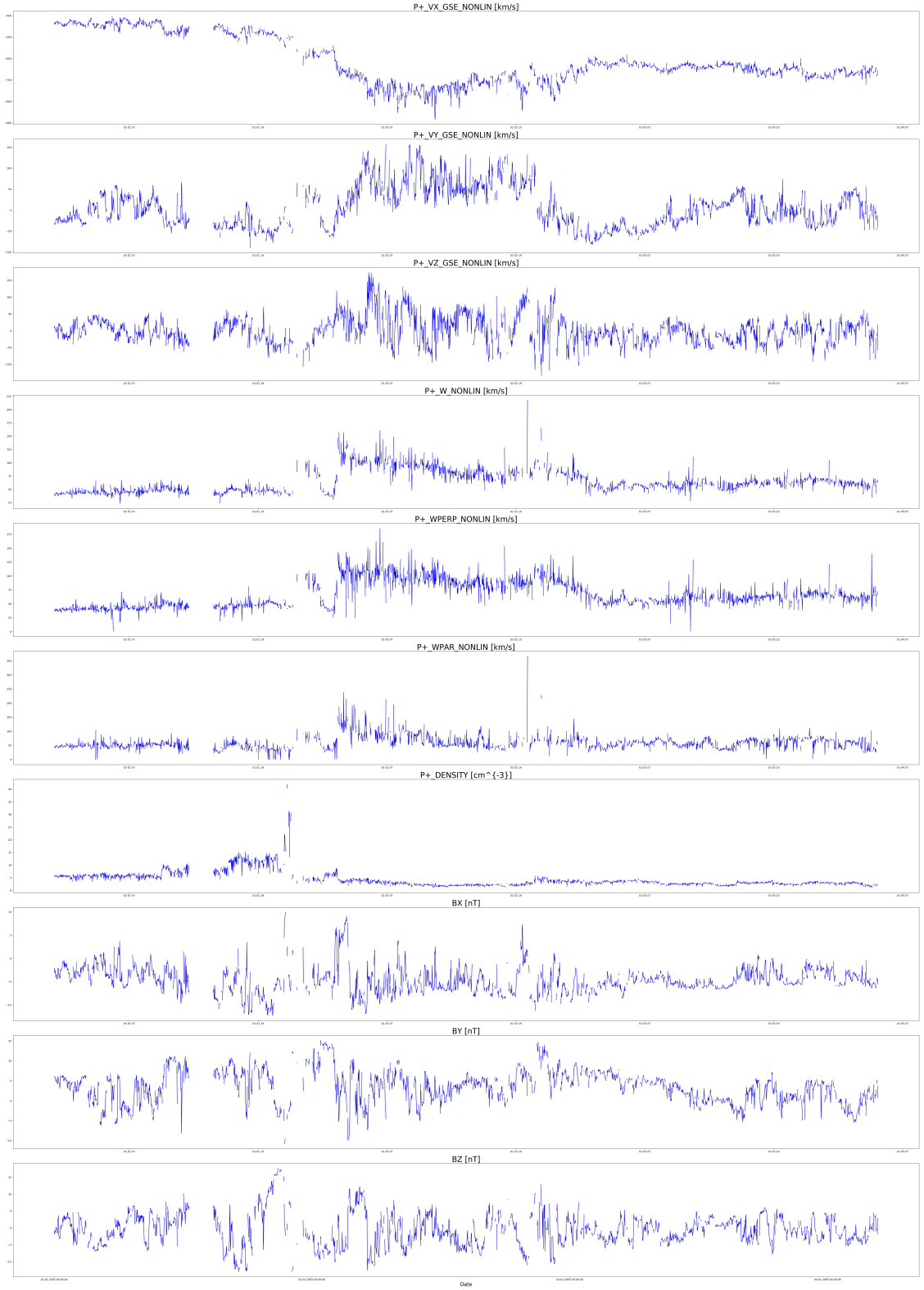


Figure 3.1: SWE components plots

3.1.2 Disturbance Storm-Time index data

The raw DST data, sampled at 1 hour, is composed of

- Date and time of the sample collection
- DST value for that instant, expressed in [nT]

We used the DST to label our samples for the classification problem since its value is the one we want to classify with our neural network models. In particular, we explored both binary and categorical classification and for this reason, we used different values as thresholds to distinguish the classes.

For the binary classification the threshold is set to -50 nT, the alarm value used from the observatory on Earth, while in case of multiclass classification the threshold values are the following:

- -30 nT: warning threshold
- -50 nT: alarming threshold
- -100 nT: destructive threshold

These values are suggested by the ASI's physicist and have been chosen consistently with physical event and their relevance, combined with the necessity of populating each class with a minimum number of events which can help the neural network to learn their patterns.

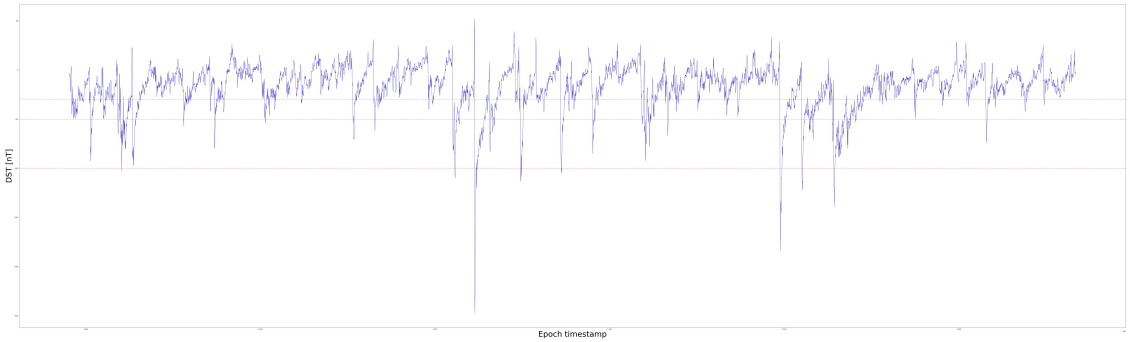


Figure 3.2: DST plot

Although all the classes are populated, both in binary and in multiclass classification, they are far from being balanced mainly due to the nature of the Sun's activity. Indeed, it is characterised from long periods of not relevant activity to reach sudden manifestations, like a CME, which does not last for long in time. Since collecting more data is not helpful in improving the quality of the considered parameters, we have to employ other methods to address the problem.

3.1.3 Class imbalance

The problem of imbalanced data typically affects classification problems where the classes are not represented equally and it can refer to both binary and multi-class classification problems. In particular, the problem occurs when there is a huge difference between the number of instances populating each class: a small difference often does not matter to the extent of training. This problem is mainly caused by the expectation by neural networks in receiving almost equally populated classes, therefore when this not occurs the model tends to make predictions biased towards the most populated class. The neural network learns too well to categorise the instances as belonging to the most populated class almost ignoring the rarest ones. In this cases we can encounter the *accuracy paradox*, a model will have an excellent accuracy value, even around 90%, but this value only reflects the underlying class distribution and it is not indicative of the model's quality. At this extent, we might end up considering a model as a good classifier, even if it is not the case, only because we do not have a correct view of the results.

The high interest in this problem comes both from the recurrences it has in many different researches and the importance of a correct classification in such sensible fields. We can take as example the cancer research in which case is really important a correct prediction, even if it leads to diagnosing cancer to people who does not really have it, and the population of healthy and sick patients are not quite balanced. Thus the issue regarding imbalanced problems is coming to the attention of many, which are conducting a large number of studies in this subject, as He and Garcia, 2009 affirms. In their paper, they identify two types of imbalanced dataset which are of our concern

- *Relative imbalance*: it is the most common type and it is characterised by the fact that even collecting new samples and letting the populations of the classes increase, the ratio of samples constituting the populations remains the same. This situation leads to poor results not only for the difference in numbers, but for the fact that most of the times is interleaved with the dataset complexity. The union of this issues hinders the neural network in correctly learning the dataset pattern.
- *Absolute imbalance (or rarity)*: in this case collecting more data cannot help in easing the problem to the nature of the data we are sampling because the minority class will aways be limited in number. In this case the learning will be difficult and we could think about changing point of view and posing the problem as an *anomaly detection* problem, always in the case there is not the chance of collecting data for the smaller class.

In our dataset the presence of *relative imbalance* is evident when observing the composition of our classes. There is a huge amount of negative events and a small number of positive and relevant events: when adopting multiclass classification the

problem is particularly visible and the population composing the rarest class is really small compared to the other classes as we can see in table 3.1. Although it does not matter the amount of data we collect, the proportion between the number of samples populating the classes remain the same.

Classification type	**Number events**	**Percentage**
Binary		
Negative class	102529	97.47%
Positive class	2663	2.53%
Multiclass		
First class	95514	90.80%
Second class	7015	6.67%
Third class	2428	2.31%
Fourth class	235	0.22%

Table 3.1: Samples distribution in classes

Problem mitigation

Many options are available to overcome class imbalance and the majority of them can be applied to both binary and multi-class classification. The followings are some of the most used and advised methods that we adopted in this work:

- Use of different metrics from the accuracy which can not be used in imbalanced datasets resulting in the *accuracy paradox* we have talked about in the past section. This metrics can give a cross-section in how a neural network model classifies the events given, helping in understanding its real performances. They can be aggregated score, as the F1 score, or else the raw classification predictions of the model without any sort of aggregation, evaluated against the *ground truth*.
- Rebalance the classes affecting the cost function implied: change the weights the neural network value more the minor classes despite the most populated ones. In this case we are only changing the importance one class has with respect to the other, or others. While training the model will be penalized more in case of wrongly predicting the less frequent class than in the case of the most populated ones. With this mechanism it will be able to learn the rarest classes.
- Resample data applying common techniques such as oversampling and undersampling. In the case of *oversampling* the instances of the least represented

class would be duplicated, whilst in the case of *undersampling* the class most represented is undersampled, deleting some of its elements. To apply these techniques there are different methods, but the random one is usually preferred to keep the representation of the various samples equal. Both techniques can be used together to balance the dataset.

We tried using the oversampling technique on our dataset, trying to enhance the minor population, but it had the same effects of changing the weights of the cost function: the model just change its threshold for the classification of the events, the number of correctly categorised events of the smaller class rises, but also the number of elements belonging to the bigger class wrongly assigned to the smaller one. The neural network does not truly learn to classify correctly each class, it just changes the threshold probability on which deciding whether a samples belong to a class or the other not leading to the improvement we wanted. The only real viable solution is to use alternative metrics to the accuracy to have a better view on the classification process carried forward by the neural network.We discuss about them later, in chapter 4.

During the experiments we investigated these techniques but discarded other ones:

- Creating synthetic samples with the help of systematic algorithms as SMOTE. For a time series, in particular the one examined in our project, it may be risky creating new samples without an effective control on the new data created since we do not truly know which are the relationships among the features. Moreover, this is not the case in which lacking values are substituted by synthetic but potentially correct values, but it is the case in which a completely new time series is created. To address this issue a new brand new project should be started to study the best way to create new samples specifically for this problem.

- Retrieving new data to increase the number of samples we have is not possible since our dataset is purely composed by data collected by NASA's missions and we do not have other sources to take them from to add more informations on the solar cycles.

3.1.4 Missing values

The presence or absence of missing data can influence not only the accuracy of the results and inhibit the use of some machine learning algorithm, but also the statistical distribution and characteristics of parameters. In the case of models robust to missing values we could use datasets not depurated, but the neural network could have difficulties in learning the pattern of the dataset besides understanding that the missing values are not helpful in the final forecasting. The causes for missing

data can vary hugely from problem to problem and, in general, it can be caused by equipment errors and incorrect measurements.

Our data are heavily affected by missing values, thus we decided to address this problem before proceeding in any further analysis. We tested different recommended best practices to deal with them, some are most specific for time series, while others are more generic and used in different ambit and with different problems. In general, there is not a solution specifically designed for each kind of dataset we want to use as input for our machine learning algorithm, but there are many different ways in pursuing the same goal, which can be used in different cases, altogether or separately, accordingly to the scenario we are addressing.

The first approach in studying our dataset, before going further in the research, is to look at the raw data as they were given us, to have a better knowledge of its characteristics. What we observed is a huge lack of values, mainly in the dataset regarding the SWE samples. This lack of samples is due to possible problems with instruments on spacecrafts and it can assume two different forms:

- Inter-row missing: the entire sample for a particular date and time is missing, causing the absence of data about all the features.
- Intra-row missing: the sample is present for the particular date and time, but one or more features which values are substituted by a default one: 99999.9.

In this scenario we found a temporal hole in the original data, going from the 24 September 2014 to the 01 December 2014 that can be considered the biggest inter-column lack of samples. This constitutes an additional reason for which we decided to invest effort in addressing this problem since it is possible that smaller or nearly equal and also we used this temporal hole in dividing the dataset later.

An insight on how missing values are distributed can help us in understanding the extension of the issue and how to deal with it. For each feature we calculated the number of instances missing and reported them in the table 3.2, both as absolute number and in percentage with respect the total size of the dataset we have.

From the table, we can see that the percentage relative to a single feature is not relevant since it does not exceed the 5% for any feature, but summed together it reaches circa the 10% of the dataset size, becoming a significant slice of it. Moreover, the three components of the magnetic field have the same number of missing values: this is because their absence is always correlated and they miss in the same samples.

What this table does not take in account is the absence of features due to the total lack of the sample for that particular date and time, the samples forming the inter-row missings. A visual inspection of them shows that the size of the missings' window can comprehend from 1-2 samples until reaching the lack of 6 hours of

Feature	Number of values	Percentage
P+_VX_GSE_NONLIN	900	0.03%
P+_VY_GSE_NONLIN	9592	0.28%
P+_VZ_GSE_NONLIN	6120	0.18%
P+_W_NONLIN	164176	4.78%
P+_WPERP_NONLIN	54378	1.58%
P+_WPAR_NONLIN	111935	3.26%
P+_DENSITY	2	0.01%
BX	3628	0.11%
BY	3628	0.11%
BZ	3628	0.11%

Table 3.2: Percentage of missing values.

data and more, thus the percentages rise and their total becomes far more than the 10%. To explore more in deep this issue, we tried to cover the missing rows artificially, filling the temporal gaps with samples containing only missing values for each feature and for the specific date and time. The result is a dataset having rows formed by real collected data and manually inserted rows representing the missing observations. Starting from this dataset we can calculate the real number of missing values and also have a complete vision of how they are distributed along the dataset.

The table 3.3 is the result of our process and it is easy to compare its results with the ones in table 3.2. The percentages are much higher and since we are dealing with a time series, the lack of these data may deprive the neural network of important pieces of information useful to learn the characteristics of them. What is worse is the potential difficulty we could found in replacing them and the problems we could have linked to the methods used to solve this kind of problem since any of them could lead to changes in the distribution of the dataset and inhibit the neural network to learn.

MCAR, MAR and MNAR

Before choosing the right algorithm to recover missing values, we analysed our dataset to classify the kind of missingness by which is affected. Taking the results as a starting point, we can decide which is the best method to cover missings.

- MCAR: it stands for *"missing completely at random"* and represents the case in which the missing values are randomly distributed along the observations.
- MAR: standing for *"missing at random"*, it means that the missing values are not randomly distributed across observations but there might some sort of

Feature	Number of values	Percentage
P+_VX_GSE_NONLIN	683313	16.60%
P+_VY_GSE_NONLIN	692005	16.81%
P+_VZ_GSE_NONLIN	688533	16.73%
P+_W_NONLIN	846589	20.57%
P+_WPERP_NONLIN	736791	18.00%
P+_WPAR_NONLIN	794348	19.30%
P+_DENSITY	682415	16.60%
BX	686041	16.67%
BY	686041	16.67%
BZ	686041	16.67%

Table 3.3: Percentage of missing values
after filling the missing samples.

dependency from the other variables composing the dataset.

- MNAR: it stands for *"missing not completely at random"*, it describes the probability of an observation being missing depending on unobserved measurements, therefore the value of the observed data depends on information not available for the analysis, maybe a feature not sampled. This kind of missingness is called non-ignorable.

For this analysis, we implied the dataset we discussed in the past section, obtained by filling the original data with the rows of missing samples, to have a complete idea on how they influence the dataset statistics. We first tried to exclude the possibility of MCAR missing type despite Schafer & Graham Schafer and Graham, 2002 affirms the only way to test it is to acquire the values corresponding to the missing positions to correctly test the how much we are to that values through out tests. To test the hypothesis we chose Little's MCAR test first introduced by Little, 1988, one of the most common, which tests the null hypothesis that data are MCAR. If the p value is greater than 0.5 then missings can be classified as MCAR and missingness is assumed to be not important for the analysis.

Results of the Little's MCAR test show that the dataset is not MCAR since the value of p, in the 3.3 reported as Significance, is beneath the threshold value. In this case, the result is said to be significant and further analysis has to be done to classify data between MAR or MNAR type.

A way to test for MAR data is imputing the missing values with the ones available and then see if the predictions are consistent with the rest of the dataset or not. This kind of tests is classified as exploratory since they try different imputed

EM Means[a]

P_VX_GSE_NONLINkms	P_VY_GSE_NONLINkms	P_VZ_GSE_NONLINkms	P_W_NONLINkms	P_WPERP_NONLINkms	P_WPAR_NONLINkms	P_DENSITYcm3	BXnT	BYnT	BZnT
-420,27836	,488154488	-6,1604892	34,5469843	33,1788319	36,2774277	5,88917646	-,02586836	,036718605	,004926107

a. Little's MCAR test: Chi-Square = 9521,231, DF = 132, Sig. = ,000

Figure 3.3: Little's MCAR test results

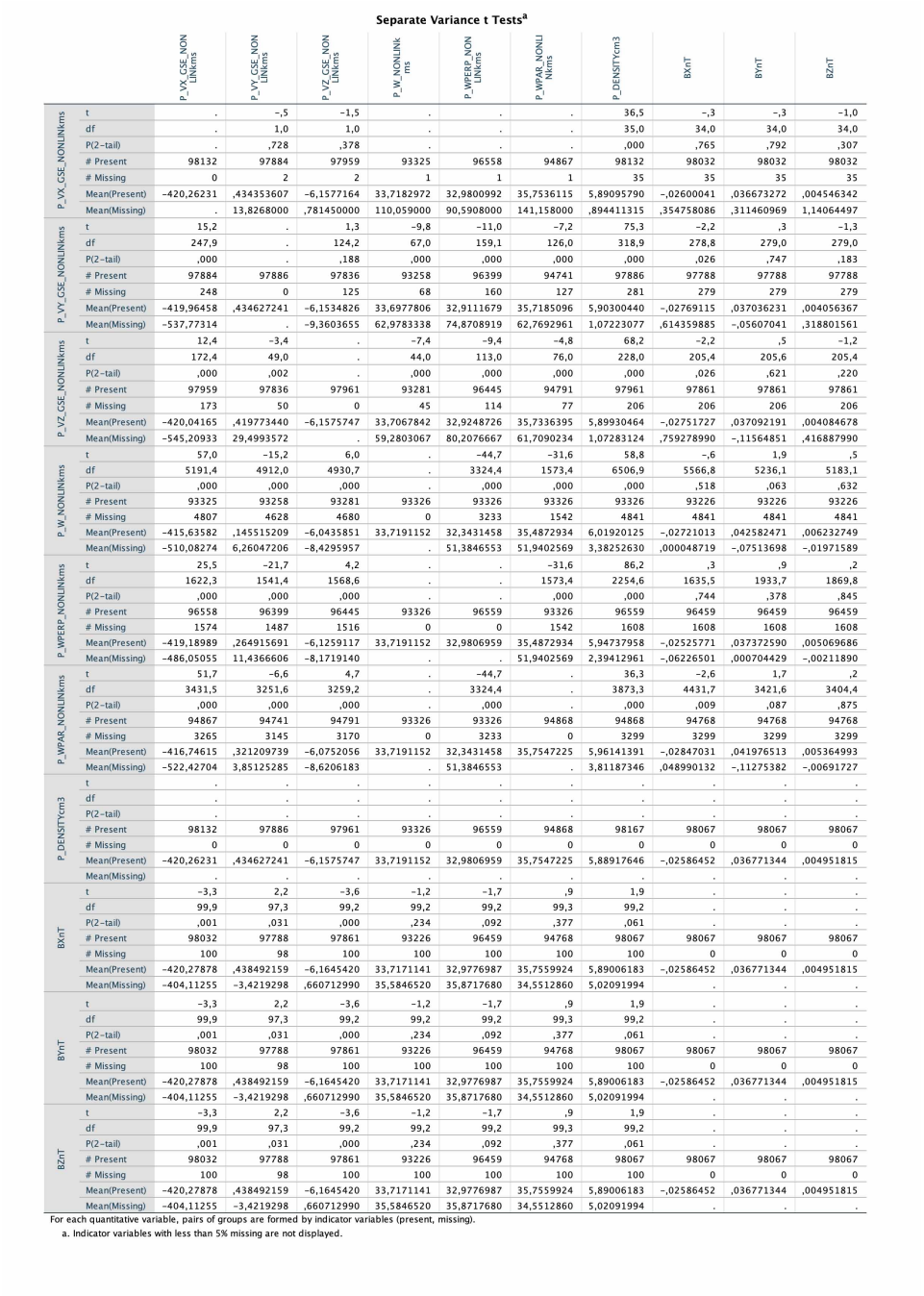
values, verifying each time which one fits better. A common statistical method used to evaluate the association between missings and the measured variable is the t-test, especially for continuous variables.

The t-test can be applied in different ambits and it is part of the family of statistical hyphothesis tests in which the test statistics follows the Student's t-distribution. It is generally used to compare two groups of variables by computing their means to assess whether they come from the same population and its statistic is based on the number of freedom degrees it can calculates from the compared groups. In the test a significant result highlight that the missingness of the features is correlated to the other measured variables in the dataset and this correlation can be used to impute the missing values other than predict their missingness. On the contrary, if data are not correlated, the missing values cannot be predicted by the present features and therefore the dataset is MNAR.

The default threshold for the t-test to estimate the missing correlation for a particular variable is a percentage of missing values above or equal the 5% and in our case all the features cover this requirement. The results are shown in the 3.4.

For the t-test to be significant the P(2-tail) parameter has to be <=0.05, meaning that the missing cases in the row variables are highly correlated to the column variable. In 3.4 the results are reported. Not all the features have consistent correlation one with the other, an example is P_VX_GSE_NONLIN which value with respect to the other features, but for P_DENSITY, is greater than 0.05. For almost the rest of the features listed, the value of P(2-tail) is low enough to let us think that the variables are MAR, thus correlated one with the other, but this it is not valid for all of them. Despite the results we cannot exclude totally the probability for our features to be MNAR, especially when we know that the missingness of data is due to external causes and we do not have enough information about the way data are collected.

Starting from this study, we can now step further, considering possible methods to recover missings and evaluating their effectiveness on data distribution and the forecasting afterwards. In fact knowing which kind of missingness affects our dataset narrows the path towards with methods to use with respect others not suitable for our problem or time series.



Separate Variance t Tests[a]

		P_VX_CSE_NONLINkms	P_VY_CSE_NONLINkms	P_VZ_CSE_NONLINkms	P_W_NONLINkms	P_WPERP_NONLINkms	P_WPAR_NONLINkms	P_DENSITYcm3	BXnT	BYnT	BZnT
P_VX_CSE_NONLINkms	t	.	-,5	-1,5	.	.	.	36,5	-,3	-,3	-1,0
	df	.	1,0	1,0	.	.	.	35,0	34,0	34,0	34,0
	P(2-tail)	.	,728	,378	.	.	.	,000	,765	,792	,307
	# Present	98132	97884	97959	93325	96558	94867	98132	98032	98032	98032
	# Missing	0	2	2	1	1	1	35	35	35	35
	Mean(Present)	-420,26231	,434353607	-6,1577164	33,7182972	32,9800992	35,7536115	5,89095790	-,02600041	,036673272	,004546342
	Mean(Missing)	.	13,8268000	,781450000	110,059000	90,5908000	141,158000	,894411315	,354758086	,311460969	1,14064497
P_VY_CSE_NONLINkms	t	15,2	.	1,3	-9,8	-11,0	-7,2	75,3	-2,2	,3	-1,3
	df	247,9	.	124,2	67,0	159,1	126,0	318,9	278,8	279,0	279,0
	P(2-tail)	,000	.	,188	,000	,000	,000	,000	,026	,747	,183
	# Present	97884	97886	97836	93258	96399	94741	97886	97788	97788	97788
	# Missing	248	0	125	68	160	127	281	279	279	279
	Mean(Present)	-419,96458	,434627241	-6,1534826	33,6977806	32,9111679	35,7185096	5,90300440	-,02769115	,037036231	,004056367
	Mean(Missing)	-537,77314	.	-9,3603655	62,9783338	74,8708919	62,7692961	1,07223077	,614359885	-,05607041	,318801561
P_VZ_CSE_NONLINkms	t	12,4	-3,4	.	-7,4	-9,4	-4,8	68,2	-2,2	,5	-1,2
	df	172,4	49,0	.	44,0	113,0	76,0	228,0	205,4	205,6	205,4
	P(2-tail)	,000	,002	.	,000	,000	,000	,000	,026	,621	,220
	# Present	97959	97836	97961	93281	96445	94791	97961	97861	97861	97861
	# Missing	173	50	0	45	114	77	206	206	206	206
	Mean(Present)	-420,04165	,419773440	-6,1575747	33,7067842	32,9248726	35,7336395	5,89930464	-,02751727	,037092191	,004084678
	Mean(Missing)	-545,20933	29,4993572	.	59,2803067	80,2076667	61,7090234	1,07283124	,759278990	-,11564851	,416887990
P_W_NONLINkms	t	57,0	-15,2	6,0	.	-44,7	-31,6	58,8	-,6	1,9	,5
	df	5191,4	4912,0	4930,7	.	3324,4	1573,4	6506,9	5566,8	5236,1	5183,1
	P(2-tail)	,000	,000	,000	.	,000	,000	,000	,518	,063	,632
	# Present	93325	93258	93281	93326	93326	93326	93326	93226	93226	93226
	# Missing	4807	4628	4680	0	3233	1542	4841	4841	4841	4841
	Mean(Present)	-415,63582	,145515209	-6,0435851	33,7191152	32,3431458	35,4872934	6,01920125	-,02721013	,042582471	,006232749
	Mean(Missing)	-510,08274	6,26047206	-8,4295957	.	51,3846553	51,9402569	3,38252630	,000048719	-,07513698	-,01971589
P_WPERP_NONLINkms	t	25,5	-21,7	4,2	.	.	-31,6	86,2	,3	,9	,2
	df	1622,3	1541,4	1568,6	.	.	1573,4	2254,6	1635,5	1933,7	1869,8
	P(2-tail)	,000	,000	,000	.	.	,000	,000	,744	,378	,845
	# Present	96558	96399	96445	93326	96559	93326	96559	96459	96459	96459
	# Missing	1574	1487	1516	0	0	1542	1608	1608	1608	1608
	Mean(Present)	-419,18989	,264915691	-6,1259117	33,7191152	32,9806959	35,4872934	5,94737958	-,02525771	,037372590	,005069686
	Mean(Missing)	-486,05055	11,4366606	-8,1719140	.	.	51,9402569	2,39412961	-,06226501	,000704429	-,00211890
P_WPAR_NONLINkms	t	51,7	-6,6	4,7	.	-44,7	.	36,3	-2,6	1,7	,2
	df	3431,5	3251,6	3259,2	.	3324,4	.	3873,3	4431,7	3421,6	3404,4
	P(2-tail)	,000	,000	,000	.	,000	.	,000	,009	,087	,875
	# Present	94867	94741	94791	93326	93326	94868	94868	94768	94768	94768
	# Missing	3265	3145	3170	0	3233	0	3299	3299	3299	3299
	Mean(Present)	-416,74615	,321209739	-6,0752056	33,7191152	32,3431458	35,7547225	5,96141391	-,02847031	,041976513	,005364993
	Mean(Missing)	-522,42704	3,85125285	-8,6206183	.	51,3846553	.	3,81187346	,048990132	-,11275382	-,00691727
P_DENSITYcm3	t	.	.	.	.	.	.	.	.	.	.
	df	.	.	.	.	.	.	.	.	.	.
	P(2-tail)	.	.	.	.	.	.	.	.	.	.
	# Present	98132	97886	97961	93326	96559	94868	98167	98067	98067	98067
	# Missing	0	0	0	0	0	0	0	0	0	0
	Mean(Present)	-420,26231	,434627241	-6,1575747	33,7191152	32,9806959	35,7547225	5,88917646	-,02586452	,036771344	,004951815
	Mean(Missing)	.	.	.	.	.	.	.	.	.	.
BXnT	t	-3,3	2,2	-3,6	-1,2	-1,7	,9	1,9	.	.	.
	df	99,9	97,3	99,2	99,2	99,2	99,3	99,2	.	.	.
	P(2-tail)	,001	,031	,000	,234	,092	,377	,061	.	.	.
	# Present	98032	97788	97861	93226	96459	94768	98067	98067	98067	98067
	# Missing	100	98	100	100	100	100	100	0	0	0
	Mean(Present)	-420,27878	,438492159	-6,1645420	33,7171141	32,9776987	35,7559924	5,89006183	-,02586452	,036771344	,004951815
	Mean(Missing)	-404,11255	-3,4219298	,660712990	35,5846520	35,8717680	34,5512860	5,02091994	.	.	.
BYnT	t	-3,3	2,2	-3,6	-1,2	-1,7	,9	1,9	.	.	.
	df	99,9	97,3	99,2	99,2	99,2	99,3	99,2	.	.	.
	P(2-tail)	,001	,031	,000	,234	,092	,377	,061	.	.	.
	# Present	98032	97788	97861	93226	96459	94768	98067	98067	98067	98067
	# Missing	100	98	100	100	100	100	100	0	0	0
	Mean(Present)	-420,27878	,438492159	-6,1645420	33,7171141	32,9776987	35,7559924	5,89006183	-,02586452	,036771344	,004951815
	Mean(Missing)	-404,11255	-3,4219298	,660712990	35,5846520	35,8717680	34,5512860	5,02091994	.	.	.
BZnT	t	-3,3	2,2	-3,6	-1,2	-1,7	,9	1,9	.	.	.
	df	99,9	97,3	99,2	99,2	99,2	99,3	99,2	.	.	.
	P(2-tail)	,001	,031	,000	,234	,092	,377	,061	.	.	.
	# Present	98032	97788	97861	93226	96459	94768	98067	98067	98067	98067
	# Missing	100	98	100	100	100	100	100	0	0	0
	Mean(Present)	-420,27878	,438492159	-6,1645420	33,7171141	32,9776987	35,7559924	5,89006183	-,02586452	,036771344	,004951815
	Mean(Missing)	-404,11255	-3,4219298	,660712990	35,5846520	35,8717680	34,5512860	5,02091994	.	.	.

For each quantitative variable, pairs of groups are formed by indicator variables (present, missing).

a. Indicator variables with less than 5% missing are not displayed.

Figure 3.4: T-test results

Methods to handle missing data

Following the considerations above and attesting the MAR hypothesis, we can move forward choosing which best practices apply to our dataset. After the application

of each method, we will test the outcome to know which one is the best choice. In particular, we evaluated as applicable to our dataset the following techniques:

- *Listwise deletion*, the simplest approach, the rows containing missing values are removed. This method creates a dataset without missing values, but the number of data to give as input to the neural network is highly decreased and it can damage the quality of our predictions.

- Using an *imputing method*, such as replacing the missings with a particular value, which can be a computed one as mean or median of the features or a value belonging to the domain of the particular feature, but not in the range to its possible values. Any of this imputing methods will need to be performed on the future datasets used as input for the model.

- Adding a *masking layer* to the neural network, a solution halfway between listwise method and forcing the neural network to learn that a particular value stands for missing.

- Methods more specific for time series, which in general convey more accuracy in the prediction of the missing samples, for example

  – Interpolating data using linear or spline interpolation, it works well when adjacent observations are similar and when there is not a strong seasonality in the dataset

  Between the methods specific for time series we also read about Last observation carried forward (LOCF) and Next observation carried backwards (NOCB), but they have been proven to introduce biases by Lachin, 2016 since, as they affirm

  > "*The only condition where LOCF is unbiased is when [...] the data used as the basis for the LOCF imputation has the same distribution as does the unknown missing data. Since it can never be proven that these distributions are the same, all LOCF analyses are suspect and should be dismissed.*"

  Moreover, this methods are effective only when the missing data are sparse along the dataset, but the huge gaps in our data may lead to many synthetic and incorrect values influencing negatively the forecastings made by the model.

We have also to remember that when a method is used, it has to be applied to all the datasets used as input to the neural network, not only the one used for training.

Listwise deletion

Implying this method, we decided to go through the entire dataset and remove every row containing one or more missing values. The result is a dataset where intra-row missing is handled, but we will have a bigger inter-rows hole. From this starting point we have two options, take the samples as they are and use them as input of the neural network model, or decide to resample them to try to avoid the holes created.

In the first case, the creation of batches to give as input to the neural network leads to batches of different dimensions, not accepted from neural networks demanding same-size ones. The only way to overcome this condition is filling each batch with empty rows, note as zero-padding, but since we are dealing with a time series this padding cannot be added to the end of the valid data, instead it must be inserted punctually inside the sample, to not change the time correlation of the series. Moreover, this choice may lead us to the original problem, inserting new missing values, instead of definitively deleting them. The request in having same-size batches could take us to the point where no batch can be created due to the incapacity of finding enough contiguous data to fill them, leaving us with no data to train the model.

The latter solution is the one used in our experiments and takes into consideration the frequency to which the datasets are collected. Since the DST has a frequency of 1 hour while the features of 92 seconds we can decide to undersample them. To do so we first apply the listwise deletion and then for each DST value we consider the date and time of sampling as reference. Around it we consider a small window of time, going from 10 minutes before it was sampled and the punctual moment. For each window we check whether there are one or more SWE rows in the interval, in case more are present the nearest to the DST sampling time, temporally talking, is chosen.

The choice of using a window instead of the exact value to select the features samples comes from the different times of sampling and from the attempt to cover the samples missing while not creating new ones, assuming that in 10 minutes the features' values do not change too abruptly and knowing that the lack of samples for hours is rare inside the dataset, especially when choosing only 1 sample every hour. For the same reason we decided to apply the listwise deletion before undersampling, otherwise we could introduce manually rows with missing features, causing the creation of temporal holes even after the application of the method.

Interpolation

The interpolation method is the second approach we used to overcome the inter-column lack of samples. The interpolation method applied is the *cubic spline interpolation*, the most commonly used thanks to the smoother interpolating polynomials and the smaller errors compared to other interpolating polynomials functions, besides, it is more reliable since it tends to reduce the oscillations between data points. In particular, cubic spline interpolation is the only method posing constraints on the continuity of the first and second derivative, to have smooth connections between the polynomials computed based on two points.

To compose the new interpolated dataset as new time series, we used the function *interp1d* of the Scikitlearn library which accepts as input the new dates and times created at the chosen frequency and the value indicating the missing feature: 99999.9, in our case. With this method, we overcome the presence of both inter and intra column missings by generating new, equally spaced samples. Tests were done to decide which interpolation frequency is the best to correctly describe the features with the minimum loss of information.

For first we decided to use as frequency the same of the starting features, 92 seconds since our goal is not to oversample them, but only to recover the values missing in the dataset. Then we tested also the interpolation at 1, 2, 3 and 4 minutes, to understand if a different frequency could lead us to big differences in how the dataset is composed. The difference in interpolating also leads to dataset with a smaller size and carrying less impact on the training performances.

If we take as metrics to compare the dataset created with cubic interpolation the mean and standard deviation of every feature, we can compare the different outcomes, starting from the results obtained from the original dataset in table 3.5 to the inteporlated datasets in tables 3.6.

Initial dataset

	P+_VX_GSE_NONLIN [km/s]	P+_VY_GSE_NONLIN [km/s]	P+_VZ_GSE_NONLIN [km/s]	P+_W_NONLIN [km/s]	P+_WPERP_NONLIN [km/s]	P+_WPAR_NONLIN [km/s]	P+_DENSITY [cm^{-3}]	BX [nT]	BY [nT]	BZ [nT]
count	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06	3.433794e+06
mean	-3.933813e+02	2.797534e+02	1.720974e+02	4.813184e+03	1.615948e+03	3.294301e+03	5.971795e+00	1.056261e+02	1.056951e+02	1.056562e+02
std	1.628668e+03	5.277903e+03	4.218249e+03	2.132961e+04	1.248003e+04	1.775187e+04	7.647951e+01	3.248755e+03	3.248753e+03	3.248754e+03
min	-1.153190e+03	-2.612690e+02	-2.058950e+02	2.821440e+00	2.642790e-04	1.604310e-04	7.669500e-02	-4.491250e+01	-2.988180e+02	-5.223160e+01
25%	-4.760360e+02	-1.209820e+01	-1.675360e+01	2.289330e+01	2.109333e+01	2.374710e+01	2.850280e+00	-2.425418e+00	-2.450670e+00	-1.564560e+00
50%	-3.951650e+02	-1.618575e+00	-6.003660e+00	3.161000e+01	2.923780e+01	3.390240e+01	4.534890e+00	-3.604225e-02	-4.320410e-03	2.036885e-03
75%	-3.434110e+02	1.047360e+01	4.792498e+00	4.426460e+01	4.227610e+01	4.635200e+01	7.267460e+00	2.338947e+00	2.517320e+00	1.580280e+00
max	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04	9.999990e+04

Figure 3.5: Original dataset results

The use of interpolation changes our dataset considerably influencing its distribution. The only case in which the interpolation has almost the same metrics' results is the one with 92 seconds, but not for all the features. In the case of interpolation at higher frequencies, the distribution is completely different from the original one due to the undersampling we do when creating the datasets at 2, 3

Interpolation at 92 seconds

	P+_VX_GSE_NONLIN [km/s]	P+_VY_GSE_NONLIN [km/s]	P+_VZ_GSE_NONLIN [km/s]	P+_W_NONLIN [km/s]	P+_WPERP_NONLIN [km/s]	P+_WPAR_NONLIN [km/s]	P+_DENSITY [cm^{-3}]	BX [nT]	BY [nT]	BZ [nT]
count	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06	4.116169e+06
mean	-9.695862e+02	5.824942e+03	2.526466e+03	4.970912e+02	1.705168e+03	-6.758975e+02	-5.305494e+02	-8.470719e+01	-8.271765e+01	4.003285e+01
std	1.279126e+04	5.259585e+04	2.264710e+04	4.236618e+03	1.474939e+04	6.928270e+03	5.048695e+03	1.265215e+03	1.377042e+03	9.322359e+02
min	-1.704368e+05	-1.705896e+04	-2.324087e+04	-8.801861e+03	-7.066851e+03	-8.792072e+04	-6.210654e+04	-1.734130e+04	-1.883233e+04	-7.686047e+03
25%	-4.849018e+02	-1.251291e+01	-1.730283e+01	2.259026e+01	2.082138e+01	2.285391e+01	2.707660e+00	-2.595001e+00	-2.610591e+00	-1.657050e+00
50%	-3.970899e+02	-1.331996e+00	-5.747294e+00	3.166187e+01	2.940265e+01	3.351062e+01	4.454058e+00	-8.057495e-02	-2.636309e-02	2.086535e-02
75%	-3.424597e+02	1.216330e+01	6.105779e+00	4.443194e+01	4.330019e+01	4.624900e+01	7.341739e+00	2.432884e+00	2.643044e+00	1.728179e+00
max	9.724247e+04	6.172570e+05	2.624384e+05	4.851262e+04	1.635520e+05	1.743852e+04	1.340644e+03	6.809554e+03	7.982012e+03	1.214394e+04

Interpolation at 1 minutes

	P+_VX_GSE_NONLIN [km/s]	P+_VY_GSE_NONLIN [km/s]	P+_VZ_GSE_NONLIN [km/s]	P+_W_NONLIN [km/s]	P+_WPERP_NONLIN [km/s]	P+_WPAR_NONLIN [km/s]	P+_DENSITY [cm^{-3}]	BX [nT]	BY [nT]	BZ [nT]
count	6.248343e+06	6.200867e+06	6.215891e+06	6.215039e+06	6.208994e+06	6.191983e+06	6.193661e+06	6.188676e+06	6.185228e+06	6.193217e+06
mean	1.433471e-16	-3.518391e-16	-9.019570e-17	-4.905515e-17	-1.626022e-16	-4.799390e-17	6.071951e-17	-1.927946e-16	-2.164106e-16	-1.638865e-15
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-1.475126e+01	-4.125830e+01	-2.102911e+01	-2.169717e+01	-2.762038e+01	-1.547966e+01	-1.468880e+01	-2.866825e+01	-2.913180e+01	-5.045680e+01
25%	8.258167e-02	2.069406e-02	5.296061e-02	5.167096e-02	4.145924e-02	7.394310e-02	8.213067e-02	4.165496e-02	3.991521e-02	-6.861862e-02
50%	9.024789e-02	5.178767e-02	6.369403e-02	6.273956e-02	5.625544e-02	8.181843e-02	8.964916e-02	5.981498e-02	5.866401e-02	1.157647e-01
75%	9.500158e-02	7.128230e-02	7.040042e-02	6.965677e-02	6.551391e-02	8.674937e-02	9.434300e-02	7.121322e-02	7.045745e-02	2.316069e-01
max	2.464893e-01	6.992194e-01	2.856645e-01	2.917282e-01	3.631896e-01	2.455522e-01	2.451845e-01	4.390249e-01	4.516381e-01	3.969444e+00

Interpolation at 2 minutes

	P+_VX_GSE_NONLIN [km/s]	P+_VY_GSE_NONLIN [km/s]	P+_VZ_GSE_NONLIN [km/s]	P+_W_NONLIN [km/s]	P+_WPERP_NONLIN [km/s]	P+_WPAR_NONLIN [km/s]	P+_DENSITY [cm^{-3}]	BX [nT]	BY [nT]	BZ [nT]
count	3.124171e+06	3.100433e+06	3.107945e+06	3.107519e+06	3.104496e+06	3.095990e+06	3.096830e+06	3.094337e+06	3.092613e+06	3.096608e+06
mean	-3.507942e-17	1.556284e-16	2.469111e-19	-4.380987e-18	4.690115e-17	1.885607e-17	9.370410e-18	5.746181e-17	1.055723e-16	3.769910e-16
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-1.475125e+01	-4.125800e+01	-2.102948e+01	-2.169758e+01	-2.762184e+01	-1.547957e+01	-1.468880e+01	-2.866729e+01	-2.913079e+01	-5.044797e+01
25%	8.258155e-02	2.069425e-02	5.296092e-02	5.167146e-02	4.145808e-02	7.394456e-02	8.213098e-02	4.165588e-02	3.991677e-02	-6.863694e-02
50%	9.024690e-02	5.179124e-02	6.369353e-02	6.273918e-02	5.625458e-02	8.181927e-02	8.964898e-02	5.981685e-02	5.866638e-02	1.157942e-01
75%	9.500157e-02	7.128526e-02	7.040001e-02	6.965638e-02	6.551322e-02	8.674983e-02	9.434276e-02	7.121272e-02	7.045844e-02	2.316417e-01
max	2.464978e-01	6.992992e-01	2.856799e-01	2.917443e-01	3.632330e-01	2.455806e-01	2.451843e-01	4.390330e-01	4.516463e-01	3.970154e+00

Interpolation at 3 minutes

	P+_VX_GSE_NONLIN [km/s]	P+_VY_GSE_NONLIN [km/s]	P+_VZ_GSE_NONLIN [km/s]	P+_W_NONLIN [km/s]	P+_WPERP_NONLIN [km/s]	P+_WPAR_NONLIN [km/s]	P+_DENSITY [cm^{-3}]	BX [nT]	BY [nT]	BZ [nT]
count	2.082781e+06	2.066956e+06	2.071963e+06	2.071679e+06	2.069664e+06	2.063992e+06	2.064554e+06	2.062891e+06	2.061742e+06	2.064405e+06
mean	-1.506523e-17	6.780379e-17	4.282537e-17	2.167628e-18	2.180725e-17	-1.492696e-17	3.039646e-17	6.169614e-17	8.552393e-17	6.908535e-16
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-1.475125e+01	-4.125263e+01	-2.103060e+01	-2.169881e+01	-2.762215e+01	-1.547929e+01	-1.468880e+01	-2.866719e+01	-2.913155e+01	-5.043761e+01
25%	8.258193e-02	2.070151e-02	5.295745e-02	5.166826e-02	4.145527e-02	7.394549e-02	8.213099e-02	4.165699e-02	3.991797e-02	-6.860530e-02
50%	9.024890e-02	5.179107e-02	6.369237e-02	6.273789e-02	5.625520e-02	8.182094e-02	8.964934e-02	5.981700e-02	5.866357e-02	1.157948e-01
75%	9.500129e-02	7.127812e-02	7.039811e-02	6.965444e-02	6.551274e-02	8.675069e-02	9.434249e-02	7.121217e-02	7.045758e-02	2.316429e-01
max	2.463832e-01	6.986903e-01	2.855271e-01	2.915876e-01	3.630166e-01	2.453531e-01	2.450786e-01	4.387668e-01	4.513972e-01	3.967704e+00

Interpolation at 4 minutes

	P+_VX_GSE_NONLIN [km/s]	P+_VY_GSE_NONLIN [km/s]	P+_VZ_GSE_NONLIN [km/s]	P+_W_NONLIN [km/s]	P+_WPERP_NONLIN [km/s]	P+_WPAR_NONLIN [km/s]	P+_DENSITY [cm^{-3}]	BX [nT]	BY [nT]	BZ [nT]
count	1.562086e+06	1.550217e+06	1.553973e+06	1.553760e+06	1.552249e+06	1.547996e+06	1.548416e+06	1.547169e+06	1.546307e+06	1.548305e+06
mean	4.636013e-17	2.359588e-17	3.357990e-17	4.938898e-18	-8.184579e-18	7.850875e-17	1.174742e-18	-1.701075e-17	2.466647e-17	2.662450e-16
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
min	-1.475125e+01	-4.125740e+01	-2.103023e+01	-2.169840e+01	-2.762014e+01	-1.547975e+01	-1.468880e+01	-2.866710e+01	-2.912878e+01	-5.044486e+01
25%	8.257663e-02	2.068551e-02	5.295475e-02	5.166435e-02	4.145390e-02	7.394089e-02	8.212804e-02	4.165138e-02	3.991338e-02	-6.858847e-02
50%	9.024875e-02	5.179480e-02	6.369367e-02	6.273946e-02	5.625706e-02	8.181984e-02	8.965069e-02	5.981799e-02	5.866783e-02	1.157634e-01
75%	9.500178e-02	7.128914e-02	7.039971e-02	6.965598e-02	6.551412e-02	8.674966e-02	9.434327e-02	7.121412e-02	7.045819e-02	2.315112e-01
max	2.464991e-01	6.993940e-01	2.856885e-01	2.917538e-01	3.632177e-01	2.455832e-01	2.451940e-01	4.390610e-01	4.516233e-01	3.967572e+00

Figure 3.6: Interpolation datasets results

and 4 minutes and the oversampling that takes place when applying the 1 minute

interpolation.

Since we will apply this method to all the datasets given as input, we are sure the neural networks will learn the new representation and will be able to classify never seen data, but this method could introduce new relationships between the features, not present in the initial one. In this optic, when we will analyse the results of the neural networks we have to take into account the possible bias deriving from the method.

We decided to apply interpolation to our dataset also to have better performances with the neural networks since most of them are slow to train, and applying this technique could help us in making the process faster, especially when using frequencies as 1, 2, up to 4 minutes without loosing too many samples.

Imputing

Applying this method to our dataset is particularly simple since it means to not delete the values which mark the missings due to the fact that they are already a value in the domain of our features, but not belonging to the ones that can take from them.

The data needs, hovewer, to be elaborated before being given to the neural network due to the inconstant sampling frequency time. One approach used is to fill the dataset's temporal holes with synthetic rows containing only missing values for all the features and then to undersample the rows at 1 hour, as we discussed above, but without implying the listwise deletion. In this way, the neural network will have to learn the meaning of the missings and learn to ignore them.

In this case we take advantage of the robustness of the neural networks in understanding whether a value can be considered as non important for the ultimate goal to which the model is preposed. Another option is to avoid undersampling and use the dataset, after filling the missing rows, as it is, but in our case it would be extremely difficult to create batches of the same size, as we discussed for the *listwise deletion*.

Later in the chapter we will discuss the outcomes of interpolating on the dataset compared to other methods described in this sections based on further statistical analysis and also the performances of the neural networks in chapter 5.

3.1.5 Time series decomposition

To address the problems characterising our data is necessary to analyse the features that constitutes it starting from their decomposition as time series. Each time series can be described with a simple formula

$$Y_t = l_t + m_t + s_t + \epsilon_t$$

where

- l_t is the average value in the series

- m_t is the trend

- s_t is the seasonality

- ϵ_t is the noise, the random variation present in the series

The temporal series can be seen as an additive signal, as we wrote in the formula above, or as multiplicative signal, where the components are multiplied.

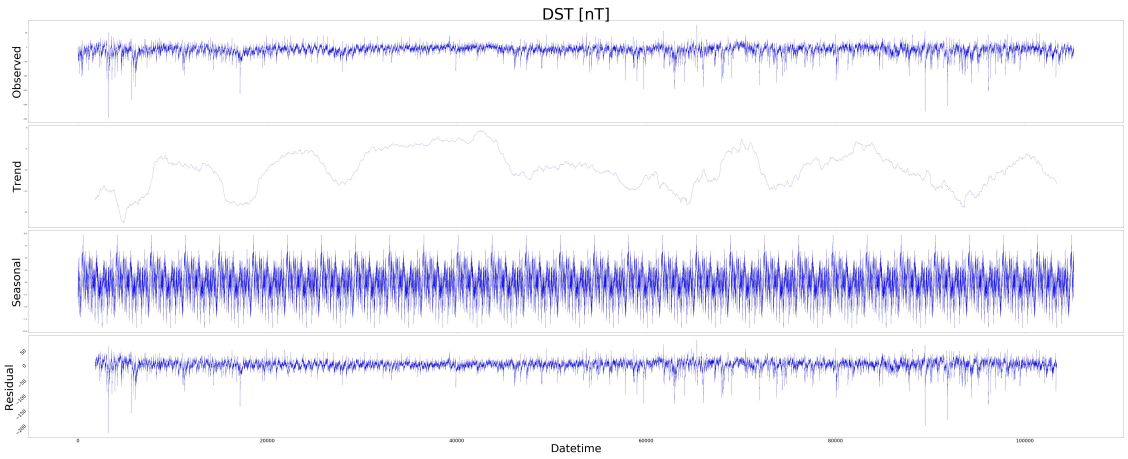


Figure 3.7: DST additive seasonality

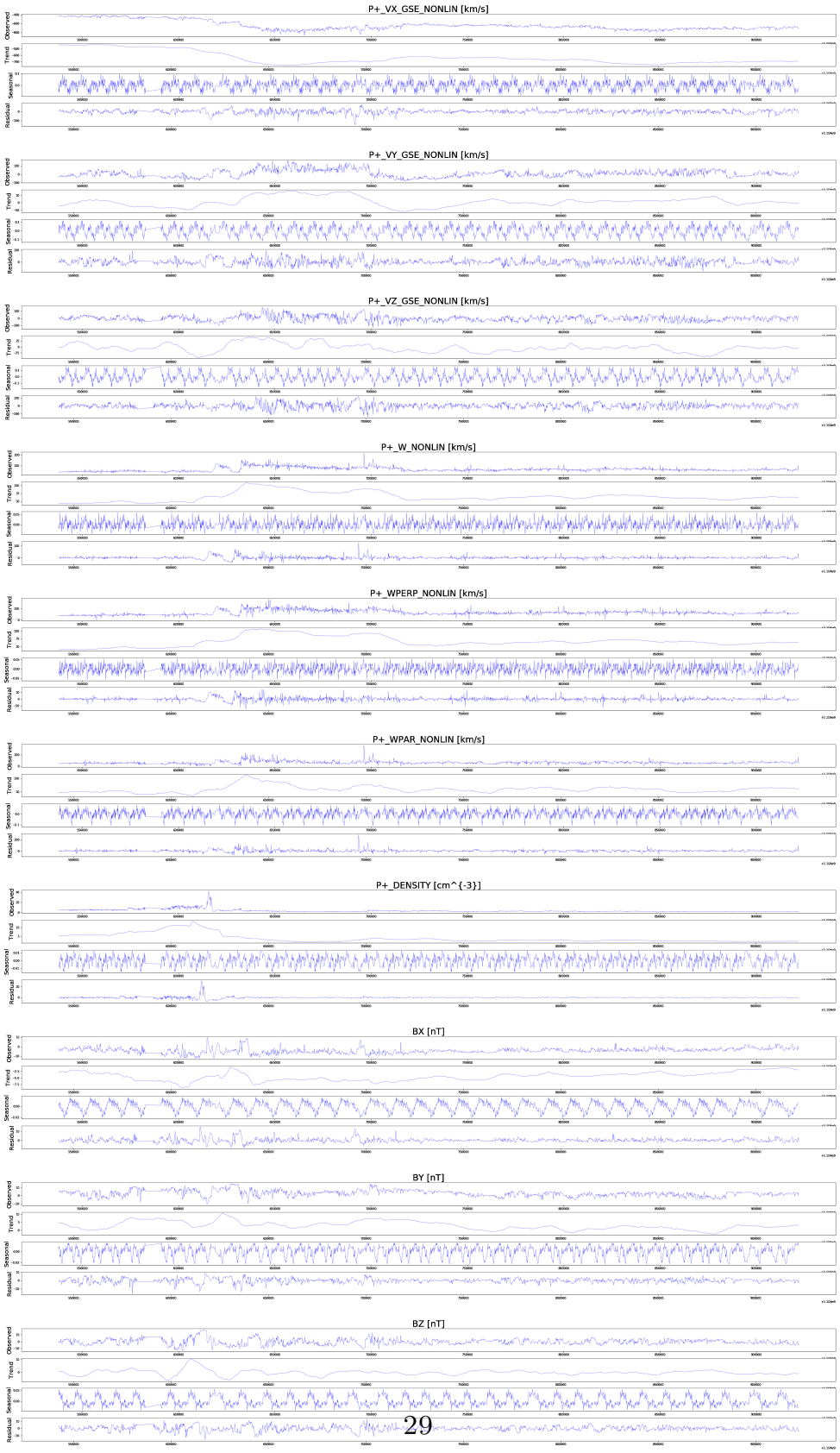


Figure 3.8: SWE additive seasonality

The graph 3.8 shows the study conducted on the SWE features. It has been computed and plotted after eliminating the rows containing the missing features present inside the dataset to avoid the possible alterations originating from them. Each feature has been decomposed and seen as a single signal, separated from the others. Aside from the gaps of missing values, highlighted by straight lines, we can observe that almost all the features are characterised by a seasonality which repeats itself, even if there are slight differences from one cycle to another. A huge example of seasonality is given by the feature BX [nT], for this feature the seasonality is evident and it has a cyclicity of circa 2 hours. On the other side, we can see that the trend is not monotonic, following the non monotonic nature of the phenomena's occurrences.

We also analysed the DST seasonality and figure 3.7 shows the results. There is a clear daily seasonality characterising it, each cycle lasts almost 1 day, mostly following the Earth's revolution around the Sun. Even in this case, the trend is not monotonic, but it fluctuates, accordingly with the values of the DST itself.

Numerous are the studies conducted on trend and seasonality, most of the times in the researches seasonality is eliminated from the data gathered following a procedure called *seasonality adjustment*, used especially when seasonality could mask more important events present in the data. This process is also applied to give as input a so-called *stationary* time series, making the modeling easier, especially for statistical modeling methods.

On the other hand, we risk to delete from the features relevant information about the problem that could help in forecasting. We must add that in the case the time series has a non linear trend even after the process of seasonality adjustment, other temporal structure could be present. Thus this operation becomes iterative, needing the repetition of the process more times.

3.1.6 Methods for statistical analysis

In this section, we present the statistical parameters and metrics we used to describe the dataset and used in the comparison between the original dataset and the ones obtained after applying the methods we will see in the following sections. In this way we can understand how the alterations applied change and influence the distribution and characteristics of the raw data: our knowledge can be used to better asses the result coming from the neural networks and also try different paths searching for the one that suits better our interests.

A note needs to be done: not all the datasets are analysed with these methods since the presence of missing values, as 99999.9, alter the statistical properties.

We employed different statistical metrics, in particular the ones which may help us in discovering some sort of correlation between the features, ranging over the

most common ones to the most specific for continuous statistical variables and time series:

- Correlation: correlation measures how close two random variables are to having a linear correlation one with the other. On time series it can be applied as cross-correlation: it measures the similarity between two series as function of the displacement of one with respect to the other.
- Distance correlation: it has the same definition of the correlation, but when measuring the correlation, but it has been demonstrated that Correlation = 0 does not imply independence, contrary to Distance correlation = 0 which implies independence.

Many other metrics have been tested such as Kendall's Tau and Distance Correlation, but without adding more information to the results found with the ones listed.

Feature **Mode**	**Mean** **Standard deviation**	**Median**
P+_VX_GSE_NONLIN	900	0.03%
P+_VY_GSE_NONLIN	9592	0.28%
P+_VZ_GSE_NONLIN	6120	0.18%
P+_W_NONLIN	164176	4.78%
P+_WPERP_NONLIN	54378	1.58%
P+_WPAR_NONLIN	111935	3.26%
P+_DENSITY	2	0.01%
BX	3628	0.11%
BY	3628	0.11%
BZ	3628	0.11%

Table 3.4: Percentage of missing values.

Features correlation

Correlation is a good metric to understand how parameters are associated one with another especially when we do not have full knowledge of the physics phenomena linking them as in our work. It helps in confirming what we sense by the simple observation of the dataset and gives us hints about not straightforward relationships. The kind of correlation we described above have been computed on the original dataset and also to the augmented datasets, after the transformations we described, to compare them. To compute correlation, features must have the same length to make a correct comparison, for this reason, in almost all the test we have

done, we can not take into account the values of DST, due to the difference in sample frequency we described.

The figure 3.9 show the results of the computed correlation applied to the initial dataset and its different elaborations, we will go through them to understand how these changes influence it.

- Figure 3.9a shows the correlation applied to the original dataset, taken as it is, without deleting the missing values or applying interpolation. It confirms what we already know about the parameters: the ones known to be linked from a physical point of view show a high positive correlation even in this case Features not known to be correlated do not actually show any correlation even after this analysis and the presence of missing values do not seem to alter the analysis.

- Figure 3.9b: here the correlation is computed over the same original dataset but after deleting all the samples containing missing values: new correlations are highlighted, not only positive but also negative ones. In comparison with the preceding results, in this case correlations before highlighted are not present anymore, for example, the case of the P+_VZ_GSE_NONLIN features that before had a positive correlation with the other two components of the SWE speed while now it seems uncorrelated from all the features. Whilst some features seem now uncorrelated, some of the ones that before were positively correlated are now negatively correlated, an example is the first component of SWE speed P+_VY_GSE_NONLIN with respect its parallel and perpendicular components, P+_WPERP_GSE_NONLIN and P+_WPAR_GSE_NONLIN.

- Figures 3.9c and 3.9d describe a completely different scenario. After interpolating the features, the correlation changes abruptly and it seems as new relationships are established between features. The computation is done starting from two different interpolated datasets that are respectively the ones used in the fig 3.9a and 3.9b. Relationships and correlations not present in the dataset before applying interpolation are now present and hugely relevant between the features. From this point of view we can affirm that interpolating causes the features to change drastically and that it is similar in applying a new representation of them,

  An additional observation is about the correlation of the DST, we interpolated the DST to 2 minutes along with the other features and there is a total lack of correlation between them when the samples are considered at the same instant of time even though the value of DST depends on the features.

- The last graph 3.9e describes the relationships between the features undersampled at 1 hour, after deleting the samples containing missings, in order to

understand how the features are related to the values of DST without modifying it. In this case there are relationships both between the features and the DST, as we know that the DST value depends on the features.

From this analysis, we can affirm that missing values can interfere and influence heavily the relationships between features as we can see from the differences between figure 3.9 and 3.9b along with the usage of interpolation which changes completely them and creates a completely new representation. In addition figures 3.9b and 3.9e seems really similar and this is due to the deletion of samples containing missings before any operations, the relationships between the features does not change even in case of change in frequency of sampling.

Cross correlation

Since we are dealing with a forecasting problem, our interest is to forecast correctly the event occurring but also to anticipate them to prevent the damages they can cause. Thus our attention is directed in understanding not only the immediate correlation between the features and the values we want to predict but also their interaction and relationships through time. The analysis we followed is made with four datasets analysed in the past section, where two are the ones used also in the analysis of correlation of the past section

- Dataset interpolated at 92 seconds
- Dataset undersampled at 1 hour
- Dataset undersampled at 1 hour with missings
- Dataset undersampled at 1 hour where missings values are deleted after the undersampling operation

In figure 3.10 we can see how interpolation does not guarantee any kind of temporal correlation between the DST and the features, no matter how further we go in future nor if the compute the correlation punctually between features and DST, whilst for figures 3.13 and 3.12 the situation appears different. Here almost all the features have some correlation with DST values, both positive and negative, despite the component of solar wind velocity on the z-axis and the two components of the magnetic field on x and y axis, whose correlation values are near the zero, meaning independence between them and the DST.

From cross-correlation we can understand the temporal relationships between DST and the features, choosing how much in the future we will forecast the events to have a good accuracy, but also anticipation in classification. Figures 3.13 and 3.12 show mainly that from the current time till 8 hours in the future the correlation is almost constant for all the features, while for P+_DENSITY and BZ the correlation is high for the first ours and then slowly decreases.

For this reason, we will concentrate on our model in forecasting the first hours and comparing the results with the ones given by forecasting further in time, for example for 8 hours in the future.

Undersampling is also influenced by the moment in which the rows with missings are deleted, whether is done before or after the operation, since it can happen in our operation that a row containing missing values is selected. This values when calculating the cross-correlation influence the results as in figure 3.11, here the correlation between the DST and the features is almost zero, hiding the real relationship existing between them, in this optic is better to delete samples containing missings before any computation is performed on the dataset.

3.1.7 Dataset preparation

After analysing the data we proceeded with the creation of the datasets to be given as input to the neural networks.

Data division

In the creation of our dataset, we applied the common scheme of division in train, validation and test sets, following the 80:20 ratio commonly used in literature. In particular, we took advantage of the lack of samples in the original data, spacing from the 25 September 2014 to the 01 December 2014 to divide the train from the other two sets. In this way, the percentages of data are

- Traning set: 82.77%
- Validation set: 8.83%
- Test set: 8.40%

Following this division and the different methods for the creation of the dataset, the amount of events we want to forecast changes from the percentages we described in table 3.1, but the proportion between them remains the same.

During the creation of each dataset the *feature scaling* was applied in the form of standardization method, to have homogeneity among different features with largely different scales, letting each feature to contribute equally to the neural network model training. This process makes the values of each features have zero-mean and unit-variance and It is applied also because many machine learning methods, as the usage of objective functions and gradient descent, will not work properly. Its formula is the one reported below and its one of the many *feature scaling* techniques that we can apply.

$$x' = \frac{x - \overline{x}}{\sigma}$$

Mini-batch technique

The mini-batch technique is widely used in machine learning to let the neural network model update its weights before all the dataset has been passed as input, in this way the model is faster in learning and reaches better results in less epochs. We used this technique to give at each iteration to the neural network a window of data to consider to classify the events, this is a way to mimic the observation done by researchers that take in account an amount of the past samples to understand which will be the future outcome. The different sampling frequency has been decisive in choosing this method, especially in the case of interpolation at 92 seconds where there is not the possibility to have a one-to-one relationship between the samples and the corresponding label.

The windows size in our case, called *batch size*, is decided considering how much we want the neural network model to look in the past to do a proper classification. We chose principally four sizes having a significant temporal meaning to us:

- 6 hours
- 12 hours
- 24 hours
- 48 hours

For each window, we tested the response of the neural network to understand which one is the best to learn and correctly forecast the events, especially because each window has a different number of samples from which the model can learn and make predictions. We decided to have so many different windows length mainly for two reasons:

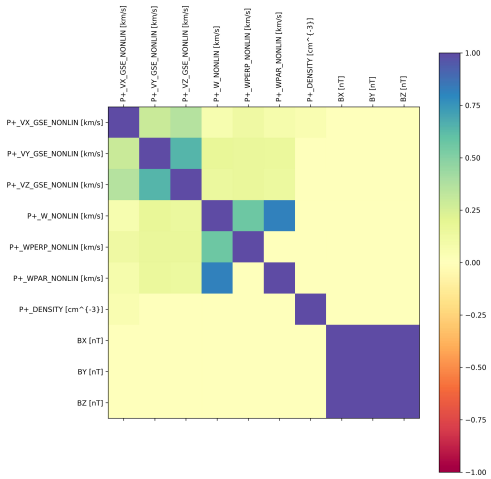
- We wanted to understand the amount of information the neural network needs to make a good classification of the events
- In the case we are using the dataset undersampled at 1 hour, a window of 6 hours means we have batch composed by only 6 samples, which seemed to us a small amount of data compared to the needed one and to the number of samples composing the batch size in the case of the dataset of interpolating at 92 seconds thus we wanted to address the issue.

Dataset created

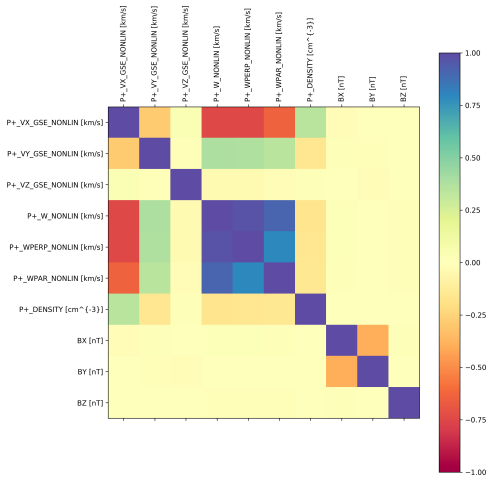
At the end of the analysis done in this chapter, the final datasets created to be used in the experiments are 4 and they are used amongst all the neural networks described in the following sections. For each of them we will repot the amount of events and the size of the mini-batches created for each window considered.

- Dataset without missings and undersampled at 1 hour
- Dataset without missings and undersampled at half an hour
- Dataset with missing data and undersampled at half an hour
- Dataset interpolated at 92 seconds

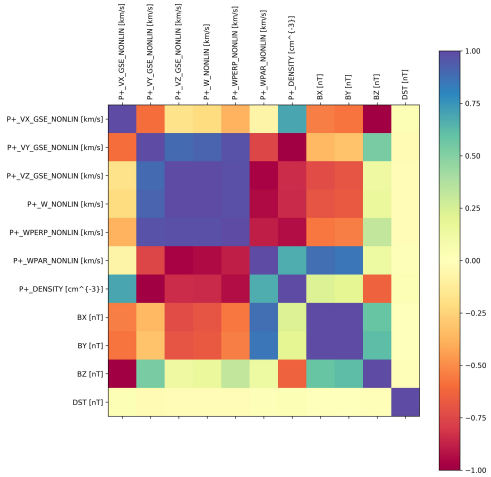
These are our starting point for the studies we will conduct and after choosing the best one we will apply different adjustments to reach better results.



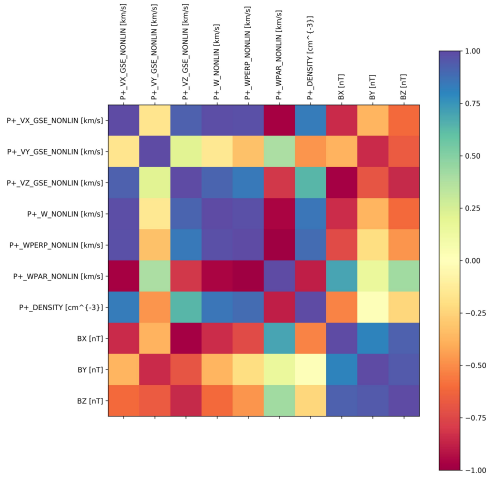
(a) Features with missing values



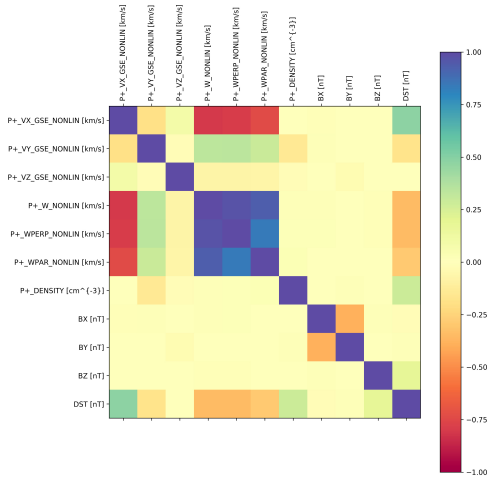
(b) Features without missing values



(c) Interpolation at 92 seconds



(d) Interpolation at 2 minutes



(e) Features undersampled at 1 hour

Figure 3.9: Features correlation

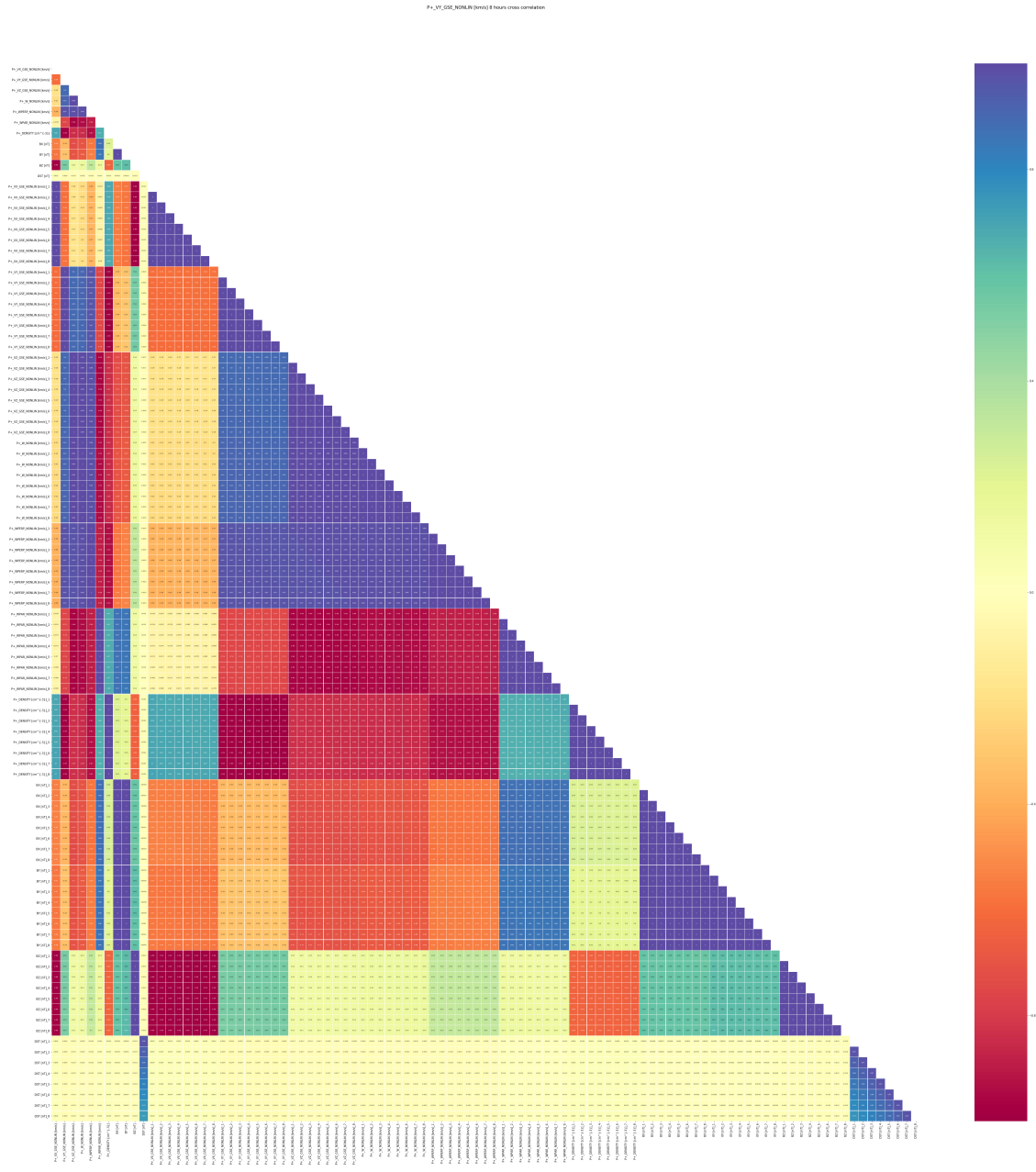


Figure 3.10: Cross correlation between interpolated at 92 seconds dataset

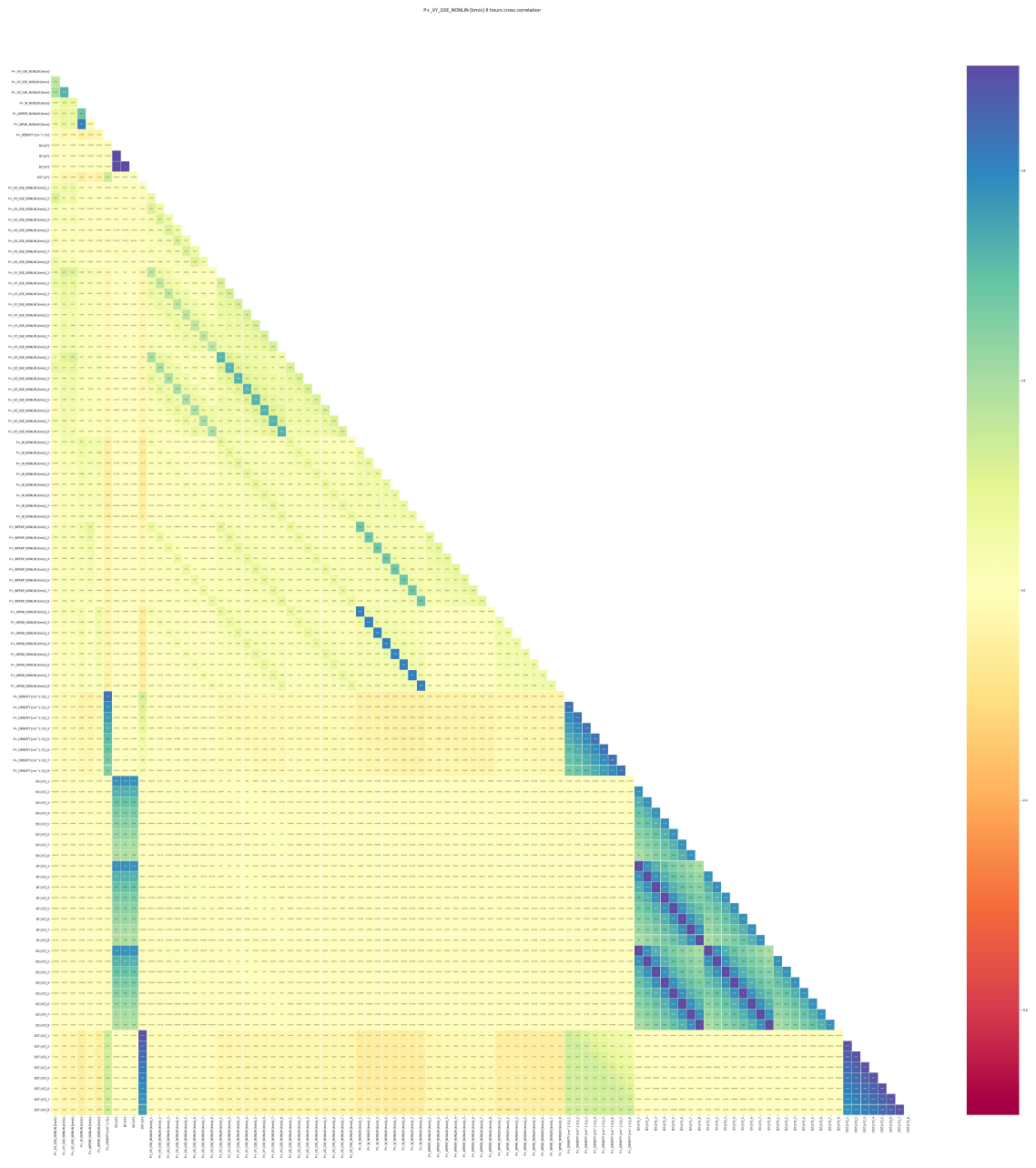


Figure 3.11: Cross correlation applied to undersampled at 1 hour dataset with missings

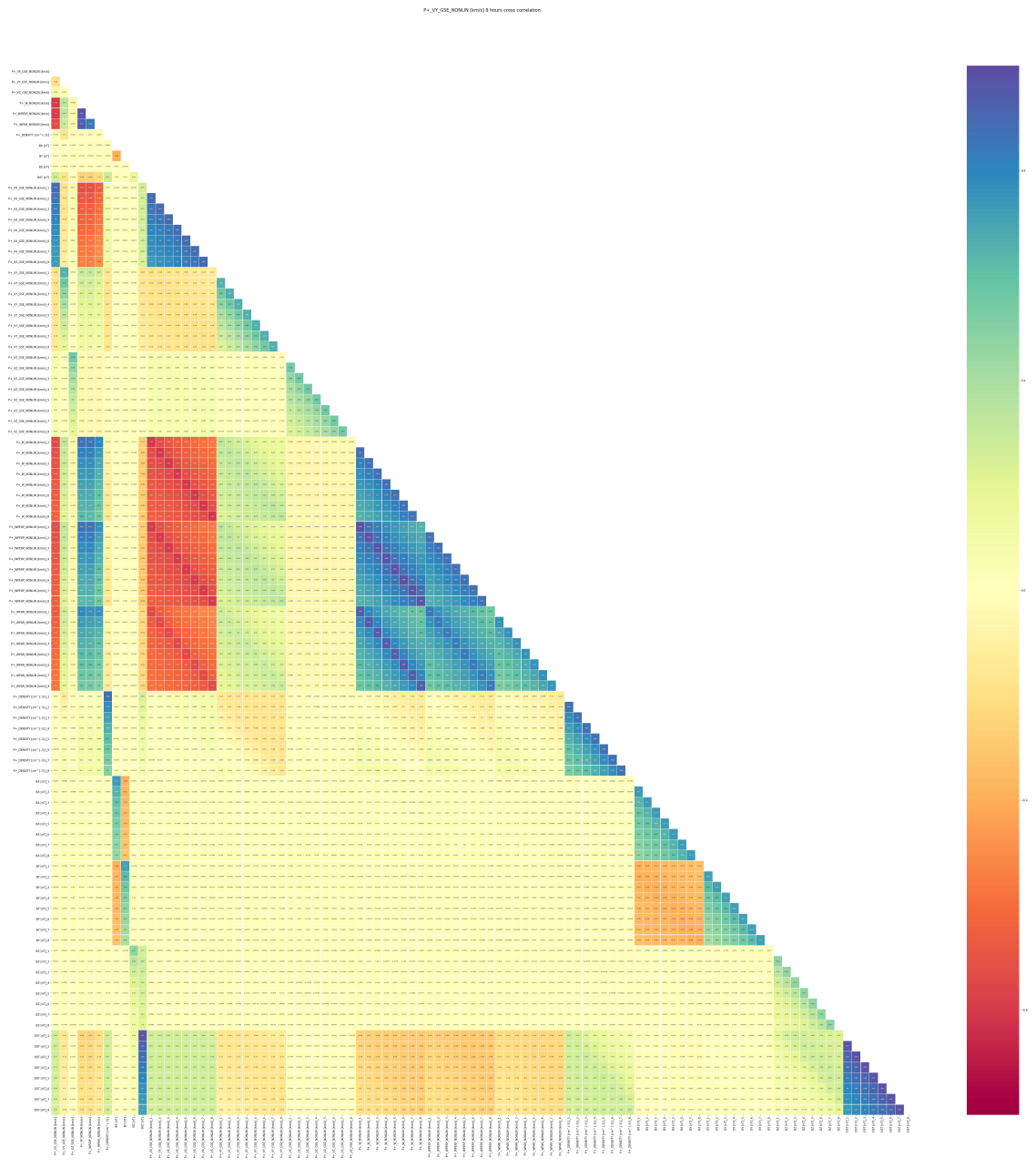


Figure 3.12: Cross correlation applied to undersampled at 1 hour dataset, missings deleted after undersampling

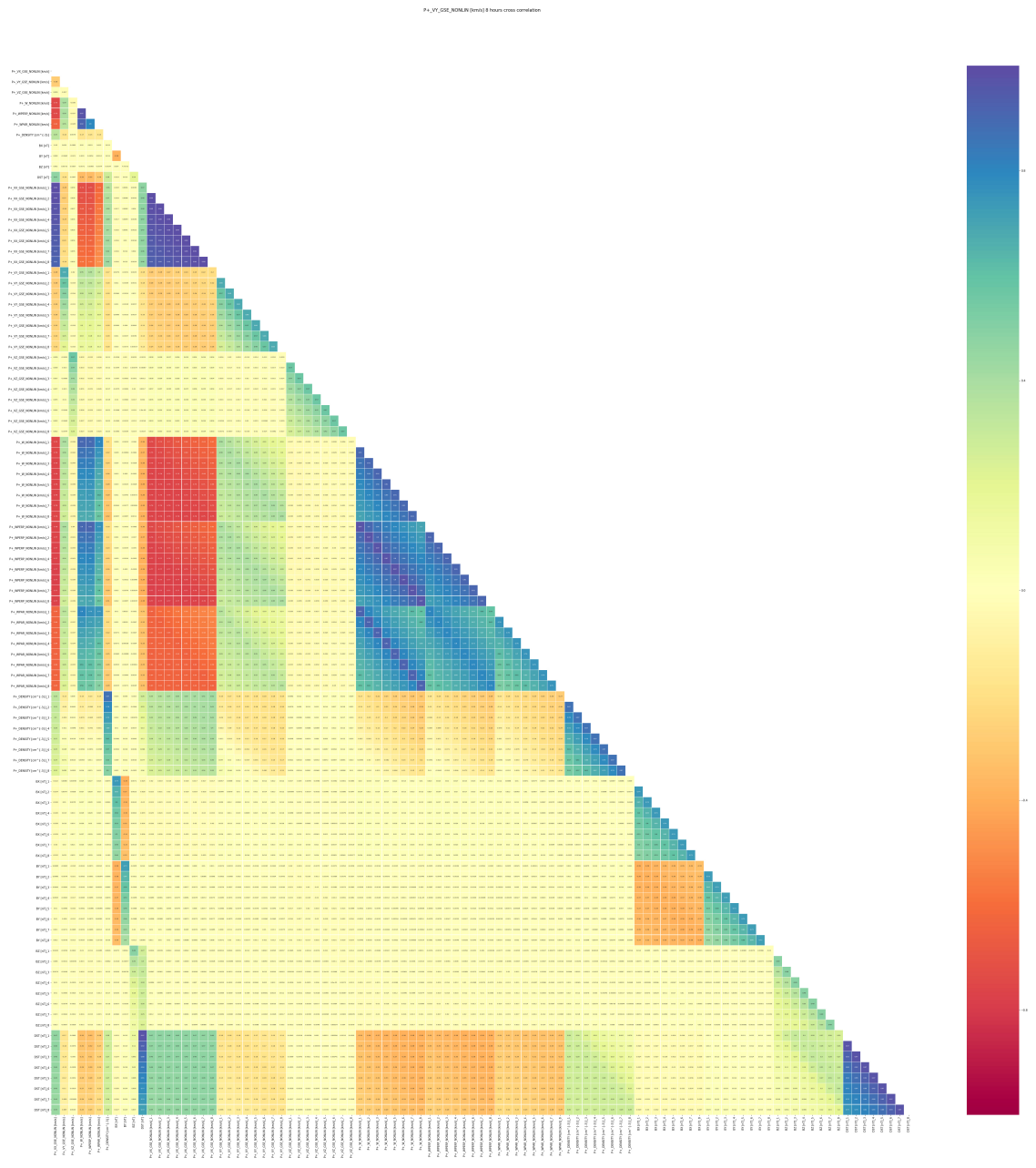


Figure 3.13: Cross correlation applied to previously depurated undersampled at 1 hour dataset

Chapter 4

Proposed deep learning architectures

This chapter will go through the hardware employed in the thesis and the software architectures used to conduct experiments together with analysing their results and the statistical analysis performed. Furthermore, we will introduce the neural networks employed in this work and the architectures realised for our project.

4.1 Hardware and software environment

What follows is a description of the workstation used, from both the hardware and software point of view.

4.1.1 The machine

All the work has been done on AMERICA, a workstation of the Politecnico di Torino. This machine is comparable to a high-end machine for video rendering/gaming applications. The machine is composed of:

1. HDD: 1.3 TB
2. RAM: 11 GB
3. CPU: Intel® Xeon® E5640 - 2.67 GHz
4. GPU: NVIDIA GTX 1080 Ti: 12 TFLOPs, 12GB VRAM

4.1.2 Software environment

The experimental server AMERICA has been the machine used for the development of deep learning models. The access to it can be via ssh or using the Jupyter Hub

interface, other than physical. While the ssh connection is suitable for maintenance tasks, the development for its entiery is performed using the Jupyter Hub Interface because it provides a more comfortable interface.

4.1.3 Scientific stack

The major components of the project are Python and Jupyter, running on Ubuntu, the Linux Operating System. Python is a general-purpose programming language, while Jupyter is a server-client application that allows editing and running notebook documents via a web browser. Jupyter Notebooks are documents that contain both code (e.g., python) and rich text elements (paragraph, equations, figures, links). Notebook documents are powerful because they are both human-readable documents that contain descriptions and results, as well as executable documents which can be run to perform computations.

All the work is based on Python 3.5.2, which is the language of choice of the deep learning community and it has solid scientific libraries. This allows us to experiment solutions and iterate extremely fast. It is to be noted that almost all Deep Learning and scientific libraries use Python exclusively as an interface, in fact, the majority of the code is written in C, C++, and Fortran for speed. The main Python libraries used in this work are:

1. SciPy McKinney, 2010: A Python-based ecosystem of open-source software for mathematics, science, and engineering.**iosresize**

2. NumPy van der Walt et al., 2011: The fundamental computing library of Python. It adds support for large, multi-dimensional arrays and matrices, along with an extensive collection of high-level mathematical routines.

3. Scikit-learn Pedregosa et al., 2011: A machine learning library. It contains various classification, regression, and clustering algorithms and is designed to interoperate with NumPy and SciPy.

4. Pandas McKinney, 2010: A data manipulation and analysis library. In particular, it offers an interface to organise and manipulate tables and time series.

5. Matplotlib Hunter, 2007: A library to plot 2D graphics.

The IBM SPSS Statistics software IBM Analytics, 2018 has been chosen, instead, to compute the statistical studies seen in chapter 3. This software is able to reliably take care of all the processes and statistical pipeline of gathering, analysing and displaying analysis performed on dataset and thanks to the many instruments available is one of the most common tools in this ambit.

4.1.4 Deep learning stack

Many are the deep learning libraries developed in the last years and most of them perform the same set of operations, but each one preserves its strength point. Keras and TensorFlow are predominantly used in this work to create models and test the results.

Tensorflow

TensorFlow is an open source software library for numerical computation that uses data flow graphs and nowadays it is the most common framework used for deep learning. Nodes in the graph represent mathematical operations, while the graph edges represent the tensors transferred between them. Tensorflow allows balancing of the computation on one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

Keras

Keras is a high-level neural network API compatible with the TensorFlow, CNTK and Theano backends. Its primary goal is to help in fast experimentation. It was chosen because it is the "lingua franca" of deep learning, meaning that we can define a model with Keras and then convert it to different frameworks that may work better in production.

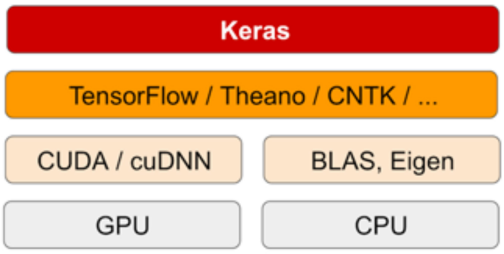


Figure 4.1: Keras software and hardware stack, adopted from Chollet, 2017

4.2 Experimental methodology

In this section, we will explain the experimental methodology we followed to make sure that the state, setup and experimental conditions were the same on all models used. Moreover, we will present the strategy and mathematical means we used for all measures performed in our study.

4.2.1 Experiment reproducibility

Experiment reproducibility is essential because it allows tracking and replicating results, enabling more detailed analysis on the reasons of success and failure of a model, while guaranteeing a proof of work for the client. In a machine learning scenario characterized by its iterations, full reproducibility is achieved only if all the steps are constantly tracked. To be more precise, it is necessary "tracking the steps, dependencies between the steps, dependencies between the code and data files and all code running arguments"**dvc**.

On the contrary, in the exploratory phase, such a protocol is not necessary, and it would eventually slow the whole process without clear advantages. Nonetheless, during the definition of the model, we may need reproducibility for a limited time. These moments include the situations where we have to run experiments to find good hyperparameter values for our pipeline. In such situations all things but the target hyperparameter should be fixed, therefore it is not necessary to keep track of anything.

4.2.2 Resources management

In deep learning experiments, we often use all the available resources, meaning that all CPU cores and the full power of the GPU should be dedicated to a single experiment. When this is not possible, we have seen that the training time does not increase when the CPU is used for other tasks that are not particularly intensive.

For what concerns the GPU, it should be used only by a process at a time. We tried to run multiple models at the same time, and we have seen that the GPU scheduling is not predictable, and it is possible that a model starves the other one. Moreover, we have noted that when multiple models run at the same time the performances decrease severely for both of them, making it faster to train the models sequentially.

The only way to let more models run at the same time is to access directly TensorFlow's backend and assign to the particular Session a fraction of the GPU in use. This can be a good solution while exploring all the possible solutions and hyperparameters combinations, but during the final tuning of the model is not advisable to use.

4.2.3 Hyperparameter optimization

After the network is defined we proceed with the definition of the hyperparameters of the neural network. Usually we start with the default ones and after some experiments, we can choose manually reasonable ranges for the hyperparameters. It is important to note that we can make this manual adjustment because we start with good models that usually have built-in mechanisms to prevent overfitting.

4.3 Neural network models and architectures

There are numerous neural network models available for training and we decided to employ the most common and suitable for time series. In addition, the characteristics of our dataset bounded us in a way that not all the models are suitable to our intent, especially the ones in need of a one-to-one correspondence between sample and label, such as Support Vector Machines. This is due to the different sampling rate present in most of the data we are working on and our decision in structuring the input as observation windows. For our experiments we applied supervised binary and multiclass classifiers, giving both the features and the labels, called truth ground, and letting the classifier learn from data themselves.

4.3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are one of the most popular neural networks used in the study of time series and Natural Language Processing (NLP) problems. The idea behind them is to use sequential information as input and use their "memory" to store the previous computation in order to let the current output depend from present and past steps. This mechanism can be really useful when we want to make predictions based on time, especially when the model needs context to make predictions, but there are some limitation on how much they can go back, lacking the capacity to depict long-term relationships in data. This kind of network exploits Backpropagation Through Time (BPTT) as mechanism to learn and it is the Backpropagation commonly used by normal neural networks applied to an unrolled RNN which means that the error is back-propagated from the last to the first timestep, unrolling all of them, allowing the update of the weights. RNNs tend to suffer mainly of two problems regarding the gradient which makes other and more powerful neural networks preferable

- Vanishing gradient: the gradient becomes smaller at each iteration and the model stops learning or else it takes too long to do it.

- Exploding gradient: in this case the model assigns a value too high to the weights, but this problem can be solved truncating the gradient itself as for example using the technique called gradient clipping.

These problems prevent the RNNs to go too far back in time and learn much more complex patterns, for this reason we have to take in consideration other networks more advisable for our purposes and also able to bear a deeper architecture compared to the ones possible with RNNs, such as IndRNN and LSTMs.

Independently Recurrent Neural Network

The Indipendently Recurrent Neural Network (IndRNN) has been recently proposed by Li et al., 2018 which tries to overcome the known problems of RNNs and their difficulty in training. It introduces independence between neurons belonging to the same layer and allows the connection between neurons belonging to different layers, as one neuron also receives as input all the outputs of the neurons appertaining to the previous layer, allowing exploration through time. In this way, the common problems involving the gradient are avoided and a major depth of the neural network can be reached, translating in the capacity of processing longer sequences, without causing the gradient decay that affects other neural networks such as LSTM and GRU. Always in Li et al., 2018 they states that an IndRNN have a behavior similar to the one of CNNs, which can be seen as a big improvement compared to the many issues defining RNNs. These are the reasons we decided to employ this architecture with respect the classic RNN one.

The independence of the neurons allows also the parallel computation of outputs increasing the speed of IndRNNs and making them preferable to RNNs and LSTMs, much slower.

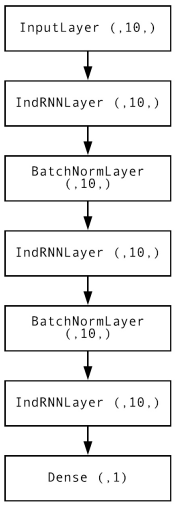


Figure 4.2: IndRNN architecture

4.3.2 Long-Short Term Memory

Long-Short Term Memory neural networks have been introduced as extension of RNNs by Hochreiter and Schmidhuber, 1997, involving principally an extension of their memory. Their memory gives the opportunity to remember the inputs over long period of times thanks to the composition of their basic cell.

LSTMs on the other hand, make small modifications to the information by multiplications and additions while it flows through a mechanism known as cell states. In this way, LSTMs can selectively remember or forget things and the information in a particular cell state depends from three different factors

- The previous cell state (i.e. the information that was present in the memory after the previous time step)
- The previous hidden state (i.e. this is the same as the output of the previous cell)
- The input at the current time step (i.e. the new information that is being fed in at that moment)

This model is also able to prevent the problems affecting the RNNs thank to its activation function consisting to an identity function, in this way the backpropagated gradient does not vanishes nor explodes, remaining costant. Now they are widely used for their characteristics and their capability of being applied to many different tasks.

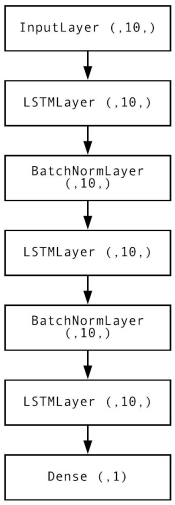


Figure 4.3: LSTM architecture

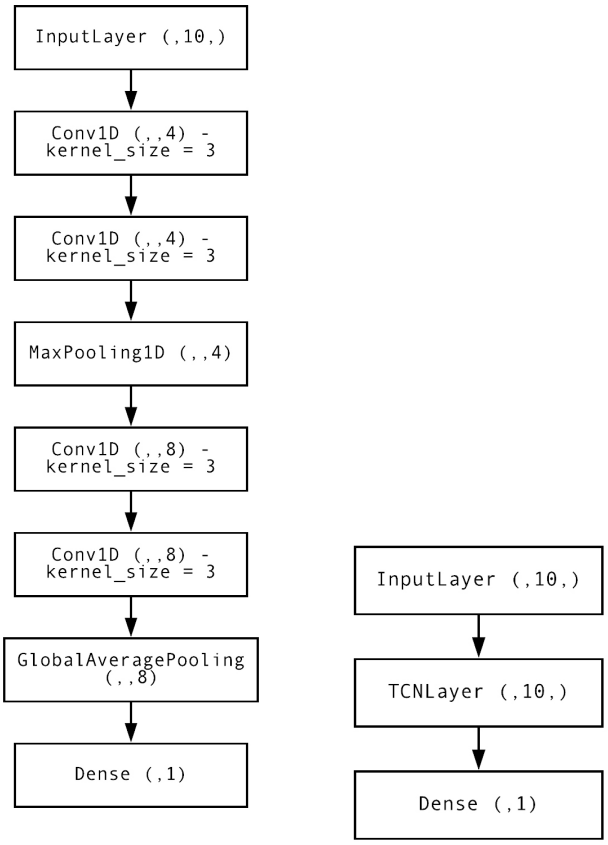
4.3.3 Convolutional Neural Network

Despite the common association between time series and recurrent neural networks, nowadays Convolutional Neural Networks (CNN), outperform really well the results obtained on the same tasks by both RNNs and LSTMs, as stated by

DBLP:journals/corr/abs-1301-3781. They conducted benchmarks on 9 common problems applied to RNNs and the results were far better in accuracy, but also in the amount of training time and the quantity of memory needed from the network.

Along with the classical CNN architecture, we also tested the Temporal Convolutional Network (TCN) family of architectures, a particular CNN created combining the best practices of the classical CNN and improving the behaviour of classical RNN. Even the usual problems affecting RNNs such as exploding and vanishing gradient are avoided, but on the other side the characteristic inherited about transfer learning is lost.

One of the downsides of such a complex neural network stands in the quantity of data required to train it.



(a) Classical CNN architecture

(b) TCN architecture

Figure 4.4: CNN architectures

What we have to underline is the necessity to tune a different number of parameters for CNNs with respect to LSTMs since some of them influences heavily

the architecture of the model. Principally we refer to the size and number of filters implied: since the datasets used do not have all the same size and in particular two of them, the ones having data undersampled at 1 and half hour, have batches of small dimensions, this parameters have to follow. In figure 4.4a, the architecture used for the datasets more populated, the one with data interpolated at 92 seconds, is reported, with a kernel size equal to 3, while in the other cases we had to use a kernel size of 1.

comment Taking advantage of the CNNs and how are easy to train, we tried to merge together the different problems of classification and regression in the case of this problem. Since with machine learning models it is possible to have a neural network running simultaneously on two separate tasks we decided to train our CNN to receive the usual input and give as output the classification value of the phenomena and also the DST value for each single batch. Even if the regression was not our first interest this gave the opportunity to the network to learn a

The TCN model in figure 4.4b seems small but only because a single layer contains a complex architecture to apply the capacity of LSTM to the Convolutional Neural Networks.

4.4 Measurement methodologies

Due to the nature of the dataset and to its composition, as we have seen before, we cannot rely only on accuracy as metric for our experiments, this decision comes from the definition of accuracy itself

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where

- TP are the True Positives
- TN are the True Negatives
- FP are the False Positive
- FN are the False Negative

From the formula it is easy to understand how is possible to end up in a situation where accuracy has an high value even if the model does not perform quite well identifying the positive class, due to the high imbalance between the classes, leading to the *accuracy paradox* we mentioned in the past chapter.

Thus, we decided to use also alternative metrics that could give us a complete view of the results given by the neural networks, some of them are more classification related, such as F1 score, Log Loss score, ROC's area under the curve, while others are more general like Confusion matrix and Precision-Recall curve.

The Confusion Matrix comes in handy especially in our case, while dealing with imbalanced datasets, since it provides a cross-section on how the predictions are distributed across the classes.

The F1 score measures the accuracy taking in consideration both *precision* and *recall*, where

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

and

$$F1score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

while the Log loss measures the performance of a classifier through the difference between the predictions made by the model and the ones expected. Since the F1 and Log Loss scores are inversely proportional, we decided to adopt only one of them, and for the amount of information that could give us and the possibility to use it in both binary and multiclass classification valuably, we leant on the first one to evaluate the future results.

We have also to underline that in the library *sklearn* the function to compute it is present, but evaluates the results in two different ways taking in consideration the two different kinds of classification: binary and multiclass. In case of binary classification the metric follow the formula reported above, while in case of multiclass classification we can chose between two different modalities: micro and macro. In case of *micro* computation, the metric is calculated globally taking into account the true and false positive and false negatives, whilst when using the *macro* modality the F1 score is calculated for every label, not taking care of the possible imbalance of the dataset determining the average. Thus, the *macro* F1 score is preferred to evaluate the results since in this case the classes have the same weight, no matter how much imbalanced is the dataset evaluated as stated by Narasimhan et al., 2016, while in the case of *micro* the importance of each class is given by the number of samples populating it, giving more importance to more populated classes.

Precision and Recall are also valuable metrics to consider, not only used in the ambit of F1 score, but also separately. *Precision* tells us how many of the classified events actually belong to that specific class, for example, in a study about cancer, it says how many of the people diagnosed having cancer, really have cancer. Whilst *Recall*, also called *Sensitivity*, measures the quantity of events part of a class are recognized by the model, in the case of people affected by cancer, it says how many people having cancer are correctly diagnosed as having it by the model. In this work, our main objective is having a higher value in Recall and a lower value in Precision since we want to correctly classify as many elements as possible in the

class of our interest, the one describing the values of DST for us relevant, despite giving value to the correct classification of a small number of them.

While F1 score, Precision and Recall are useful during the evaluation of the neural network model, the ROC and Precision-Recall curves can be used in a later stage of the work, when there is the need to balance the classes' weights and fine tune the chosen model. In this optic the value of accuracy becomes quite unimportant, but always useful to understand how a particular model is training, in the final result is possible that we will have a lower accuracy value despite the better ability of the model in classify the events.

Chapter 5

Experiments' results

In this chapter we will analyse the results of the experiments run on the neural network architectures described in chapter 4 and conducted on the datasets defined in chapter 3. Our experiments were divided mainly into three phases:

- The choice of the most promising dataset based on the results given by a preliminary analysis of a simple CNN architecture.
- The usage of the dataset chosen in the first step as input for the neural networks depicted in 4: starting from basic architectures we chose the best one to create a more complex architecture and fine tune its hyperparameters to reach our goal.
- The choice of how much in the future forecast knowing the best dataset and neural network available for this goal.

5.1 Dataset selection

For first we ran all the datasets on the CNN model since it is known to be faster than both the IndRNN and LSTM, allowing us a quick evaluation of which dataset is the best to employ. Each dataset has been used as input for binary and multiclass forecasting and the results had on the test set are listed in the tables present in this section, comparing the different windows of observation, which provide to the neural network the recent history of the features.

We also want to remember that the F1 score in case of multiclass classification is using the *macro* average in order to give unbiased results and correctly reflect the forecastings done by the model.

Even if it may seem obvious that giving to the neural network a higher number of data will lead to a higher accuracy, we have to take in account the characteristics of each neural network model since they may not benefit from a bigger batch of

data. In our case the 48 hours window is a hint of the probability a neural network may ignore data too much in the past. When it happens the gradients assume a particular form, they have a high value for the samples considered to take the decisions, the ones at the end of the window and nearer in a temporal sense, while going back in time, in the same window, the gradient decline, going really fast to zero. This means that smaller are the values of the gradient, smaller is the importance it has for the forecasting operations and this pattern can be observed with almost all the neural network architecture we created as we will see in next section. The only exception is the CNN which considers the entire temporal sample of data. The reason can be brought back to the nature of the CNN, they are born with the intent of observing an image to be able to classify them, and for this purpose, they need all the data given as input, not only the last ones.

Relative to this problem we can affirm that in every sampling method the windows giving us the poorest results was the 48 hours one: the higher amount of data is not helping the neural network in understanding the pattern beneath them. Instead, it seems the models struggle to learn and predict the classes, no matter the kind of forecasting we are dealing with. At the other extreme, we have the batches composed by only 6 hours of samples which, although the small amount of data compared to the other windows, give the chance to the network to train correctly. We have to especially underline the results given by the dataset undersampled at one hour in table 5.7, with a window of only 6 hours of data: in multiclass classification the F1 score is even higher than the ones of the other windows, and even in the other datasets its results are always better than the ones given by the 48 hours window. We can conclude that not always a vast amount of data, especially when given all at the same time, can give excellent results and for this reason, the observation window comprising 48 hours was not considered as input dataset.

Despite the good performances, the 6 hours window is not giving the best results in either of the datasets, even if comparable with the ones given by the bigger windows such as 12 and 24 hours. Thus, it will not be used for our experiments. However, the datasets undersampled at 92 seconds is the one standing out among the others, even considering this small window.

The most notable, but also predictable, results are showed in tables 5.4 and 5.5. The composition of the batches, a mix of missing values and not enough valuable samples, does not give the ability to the neural network to learn from the dataset and this is shown by the zero values of the F1 score and consequently of Precision and Recall. Even in the case of multiclass forecasting, the F1 score always has the same value, showing that the model is not actually paying attention to the input, but it always forecasts the same and most populated class. The high value of accuracy is given by the amount of false negative forecasted, as can be observed from the confusion matrix in table 5.3, here the real distribution of the predictions made by the neural network are shown. We took as an example of confusion matrix the one from 12 hours window since it seems the one giving better results on the

other experiments, but even with this dataset, the outcome is not the best one. The results concerning the other windows are not reported because are quite similar to this one.

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
6 hours	98%	0.40	0.57	0.31
12 hours	99%	0.51	0.64	0.42
24 hours	99%	0.51	0.73	0.39
48 hours	98%	0.25	0.33	0.20

Table 5.1: half hour undersampled without missings dataset - binary classification

Window	**Accuracy**	**F1 score**
6 hours	91%	0.40
12 hours	91%	0.48
24 hours	92%	0.42
48 hours	91%	0.36

Table 5.2: Half hour undersampled without missings dataset - multiclass classification

	Predicted	
Actual	8625	0
	147	0

Table 5.3: half hour undersampled with missings dataset - confusion matrix

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
6 hours	98%	0.0	0.0	0.0
12 hours	98%	0.0	0.0	0.0
24 hours	98%	0.0	0.0	0.0
48 hours	98%	0.0	0.0	0.0

Table 5.4: Half hour undersampled with missings dataset - binary classification

Window	Accuracy	F1 score
6 hours	91%	0.24
12 hours	91%	0.24
24 hours	91%	0.24
48 hours	91%	0.24

Table 5.5: Half hour undersampled with missings dataset - multiclass classification

Excluding the windows with 6 and 48 hours of samples, the windows with 12 and 24 hours of samples are the ones giving the best results, both in binary and multiclass problems. In the case of 5.6 the window composed by 24 hours gives better results, while considering the other two datasets, the window with data for 12 hours is the best one, making us leaning to this particular window size. We have to point out that overall, the results given by the dataset interpolated at 92 seconds are far better in binary classification, especially for what concerns the *Recall* that is the highest among all the datasets and being of our concern to maximise it we decided to employ this dataset in our future experiments.

In conclusion, the dataset of our choice is the one interpolated at 92 seconds with a window composed of 12 hours of observations.

Window	Accuracy	F1 score	Precision	Recall
6 hours	98%	0.40	0.59	0.30
12 hours	99%	0.46	0.69	0.35
24 hours	98%	0.46	0.48	0.43
48 hours	98%	0.0	0.0	0.0

Table 5.6: one hour undersampled dataset - binary classification

Window	Accuracy	F1 score
6 hours	91%	0.43
12 hours	92%	0.41
24 hours	92%	0.41
48 hours	90%	0.34

Table 5.7: one hour undersampled dataset - multiclass classification

Window	Accuracy	F1 score	Precision	Recall
6 hours	98%	0.43	0.62	0.33
12 hours	98%	0.53	0.61	0.47
24 hours	98%	0.20	0.37	0.14
48 hours	97%	0.07	0.10	0.05

Table 5.8: 92 seconds dataset - binary classification

Window	Accuracy	F1 score
6 hours	91%	0.38
12 hours	92%	0.46
24 hours	91%	0.40
48 hours	94%	0.36

Table 5.9: 92 seconds dataset - multiclass classification

Looking at the results listed in the tables is easy to observe how the accuracy values depict a circumstance utterly different from the ones highlighted by the other metrics, which are more descriptive of the classification made by the neural network exclusively due to the accuracy paradox we talked about in chapter 3.

In the next sections, we will use as dataset the one composed of the samples interpolated at 92 seconds with a window of 12 hours, being the one giving us the best outcome compared to the other ones. The results computed in this section concerning this dataset will be our baseline to chose the definitive model to use in forecasting the classes of events in which we are interested.

5.2 Model selection

After selecting the best dataset between the ones analysed, we proceed with the training of the architecture chosen to understand which of them is suitable for our goal. In this section we will go through the results had from running the dataset, we will evaluate their performances not only in the optic of the quality of their predictions but also considering other parameters. The models used to run this preliminary selection are the ones listed in chapter 4, they are quite simple, but they are a good baseline to understand which model to use later.

The results, both for binary and multiclass classification, for each model are showed in tables 5.15 , 5.12, 5.10 and 5.18.

The brand new model architectures such as IndRNN and TCN have outcomes way worse than the classical architectures. TCN has the worst outcomes compared

to the others, especially with respect to our baseline, even if this model has been chosen purposefully to be a CNN built for time series.

5.2.1 TCN

The TCN on the dataset selected performs poorly, taking as the point of reference the outcomes from the CNN in tables 5.8 and 5.9, our baseline. The outcomes are comparable to the ones had with the 48 hours windows. From the experiments, we run we saw how no matter the size of the window given, the interest of the networks was always focused only on the last hours of information. This peculiarity can be observed looking through the plot of the gradient with respect the input considering the different windows composing the datasets in figures 5.1, 5.2, 5.3, 5.4. The plots all refer to the binary classification, but also the ones concerning the multiclass problem are affected by the same issue. Even considering the possibility to enhance the amount of data given as input, in this case it will not be proportional to the amount of F1 score we hope for, in fact in the case of binary classification, we tried to give bigger samples windows, but the F1 score does not go further than the 0.26 in the case of the 48 hours window, highlighting a small improvement in the overall results. For this reason is better to give as input to the TCN model smaller batches, letting the neural network to learn better on the smaller amount of data we give and frequently update its hyperparameters.

The high value of F1 score for the multiclass classification is justified by the confusion matrix, the bias towards the first class is high, but the neural network can learn how the other classes are composed and to classify them, even in the case of the fourth and rarest class. The ability of the model carries to a discrepancy with the accuracy value, lower than the one computed for binary classification.

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
Binary	98%	0.16	0.20	0.14
Multiclass	87%	0.34	/	/

Table 5.10: TCN results

		Predicted		
Actual	7534	401	67	7
	495	98	20	3
	83	34	21	7
	0	0	1	1

Table 5.11: IndRNN - multiclass confusion matrix

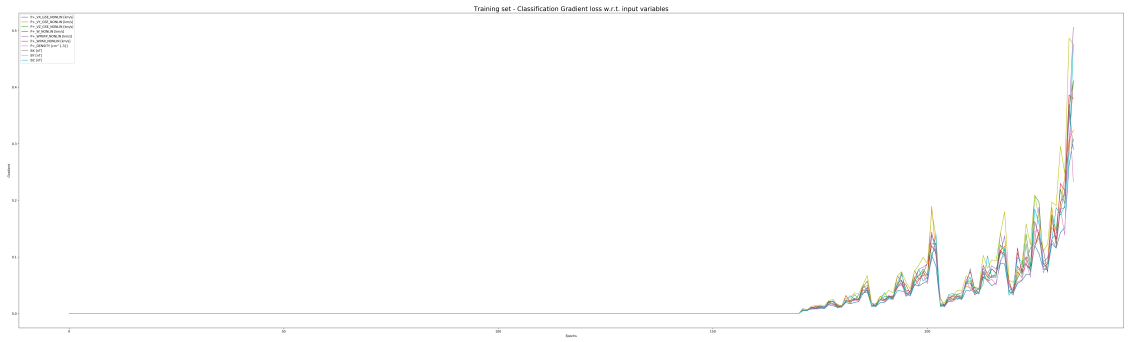


Figure 5.1: TCN - Gradient w.r.t. input - 6 hours window

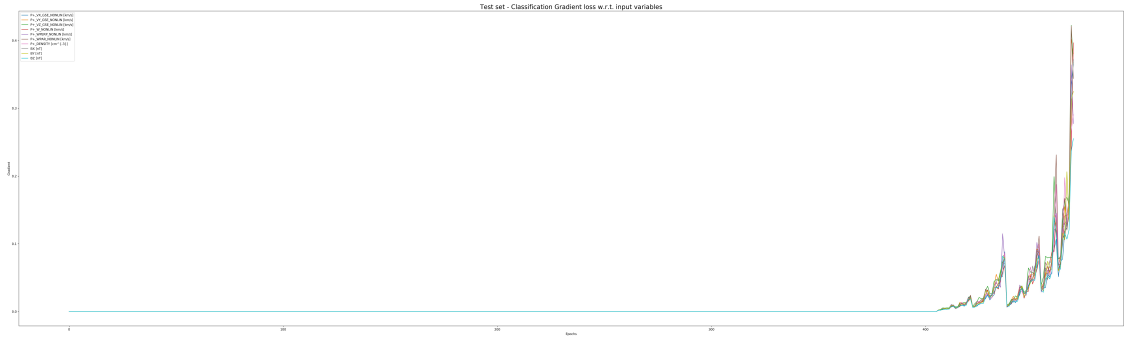


Figure 5.2: TCN - Gradient w.r.t. input - 12 hours window

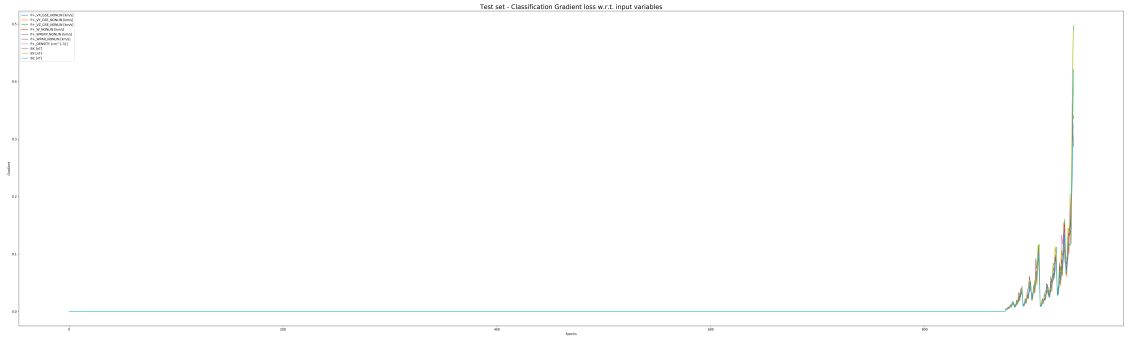


Figure 5.3: TCN - Gradient w.r.t. input - 24 hours window

5.2.2 IndRNN

The IndRNN results show a big improvement with respect the TCN, despite the gradient shows the interest of the neural network is focused even in this case on the

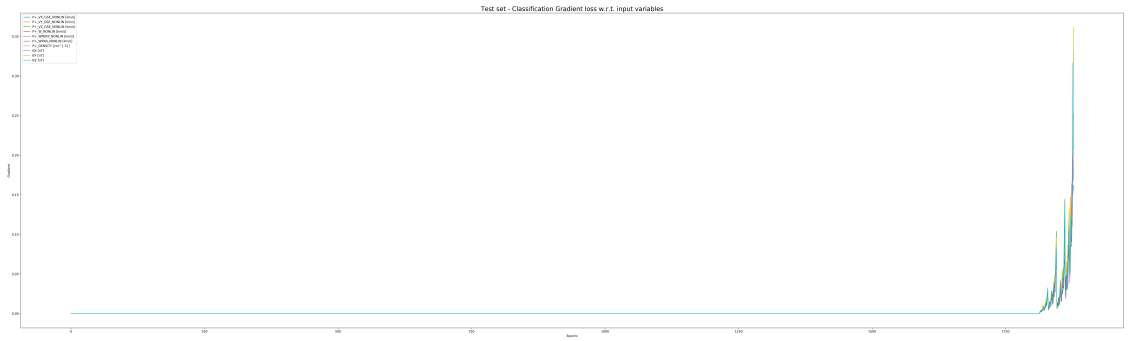


Figure 5.4: TCN - Gradient w.r.t. input - 48 hours window

hours of the entire window of samples, making the size of the window useless since to predict the class it is not taken in consideration its entirety as in figure 5.5.

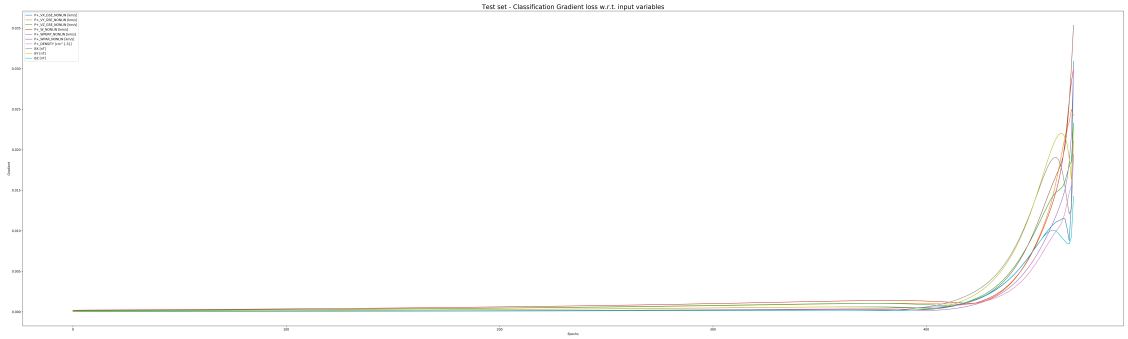


Figure 5.5: IndRNN - Gradient w.r.t. input - 12 hours window

The evaluation of this model is in table 5.12, as we announced they are far better with respect the TCN, but they do not outperform the results given by our baseline. What is evident is the Precision value which is almost nearly the perfection in the case of binary classification, but in return the Recall is almost comparable to the TCN one, lowering the overall F1 score. Being the Recall our primary interest, the high value of its counterpart makes us wonder which is the actual distribution of FP and FN while classifying the events.

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
Binary	99%	0.33	0.94	0.20
Multiclass	92%	0.33	/	/

Table 5.12: IndRNN results

Table 5.13 the confusion matrix regarding the binary classification display the

distribution of the predictions made by the neural network which are highly biased towards the negative class, having a high number of FN which influences the Precision since it does not consider them in its computation, in fact, the Recall which takes them into account is really small. In case of the multiclass problem, the value of the F1 score is the same and the confusion matrix regarding it does not improve, rather it shows the bias towards the most populated class in an amplified way, as stated by the table 5.14. Most of the samples are classified as belonging to the first class, even the ones belonging to the second and third which are more populated than the four, counting only 2 samples. What is important to observe is the second class, for which there is no correct prediction despite the higher number of samples which are classified for the most as belonging to the first class, but also as third class.

	Predicted	
Actual	8623	2
	118	29

Table 5.13: IndRNN - binary confusion matrix

	Predicted			
Actual	8002	3	4	0
	605	0	11	0
	110	0	35	0
	0	0	2	0

Table 5.14: IndRNN - multiclass confusion matrix

5.2.3 LSTM

The LSTM has better results than the IndRNN, despite the equal value of accuracy the F1 score is higher due to the higher value of Recall. This neural network has more balanced values between Precision and Recall and it is more coherent with our intent in improving Recall despite the Precision. For what concerns the multiclass problem, the F1 score is slightly lower compared to both TCN and IndRNN but looking and the confusion matrix in table 5.17, the predictions are more balanced, even if still biased towards the first class. We can see that now the second class is recognised correctly even if with some issues conversely to the IndRNN architecture, it seems like the neural network now is slightly moved towards the central classes when doing its predictions and even if most the samples are categorised as belonging

to the first class, even the third is now more considered when predicting the result. The same problem is affecting the binary classification in table 5.16, where the number of FP is considerably higher with respect the one computed for IndRNN.

The gradient in figure 5.6 has the same trend of the IndRNN's one due especially to the shared nature they have, taking into account the last observations to predict the classes for every sample.

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
Binary	99%	0.45	0.82	0.31
Multiclass	91%	0.32	/	/

Table 5.15: LSTM results

		Predicted	
Actual	8601	24
	119	28

Table 5.16: LSTM - binary confusion matrix

		Predicted		
Actual	7928	34	47	0
	572	15	29	0
	103	0	42	0
	0	0	2	0

Table 5.17: LSTM - multiclass confusion matrix

5.2.4 CNN

The CNN is our baseline of comparison between all the neural network architectures we implemented. Its results are summed in table 5.18 and they refer to ones reported in the tables of the preceding section. The results are the best we had compared to all the models on which the dataset was run, not only for the value of the F1 score that it is even better than the LSTM one but also for what concerns the Recall metric. This means that the CNN can recognise better the positive class and also to predict less false negatives, in fact, the confusion matrices in tables 5.19 and 5.20 show this ability by the CNN to classify better the other classes even if it disregards the most populated one.

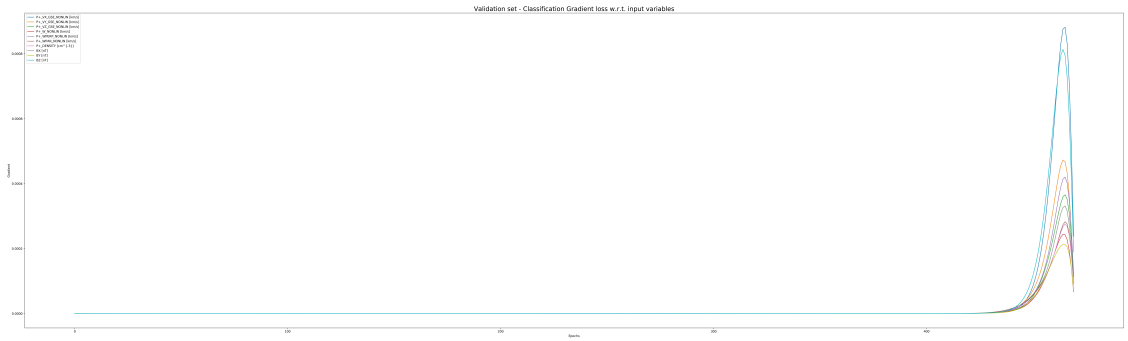


Figure 5.6: LSTM - Gradient w.r.t. input - 12 hours window

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
Binary	98%	0.53	0.61	0.47
Multiclass	92%	0.46	/	/

Table 5.18: CNN results

	Predicted	
Actual	8581	41
	78	69

Table 5.19: CNN - binary confusion matrix

	Predicted			
Actual	7859	134	16	0
	458	122	36	0
	52	34	52	7
	0	0	1	1

Table 5.20: CNN - multiclass confusion matrix

The CNN's gradient in figure 5.7 is different compared to the other neural networks since it shows that the model considers the entire window of samples to make the correct predictions giving the chance to the network to learn better from the datasets.

Overall the CNN is the neural network that is better dealing with the dataset we chose

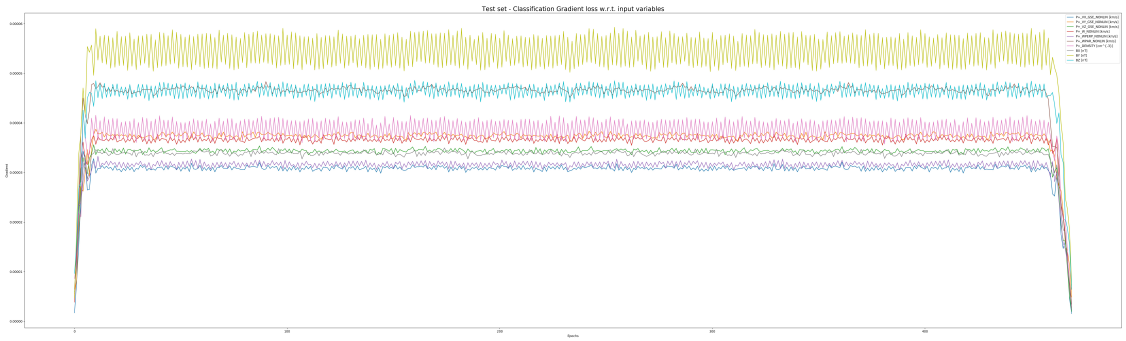


Figure 5.7: CNN - Gradient w.r.t. input - 12 hours window

The figure 5.7 refers to the binary classification, but the one for multiclass classification makes no difference, a sign that is a real peculiarity of the neural network to look at all the input given before making decisions.

This bias towards the first and third class we observed in all the neural networks we studied, but especially in the IndRNN and LSTM could be due to the division itself of the dataset. If we think about it, the first interval is composed by all the samples >=-30, the second class is composed by the ones <=-30 and >=-50 while the third is composed by the interval <=-50 and >=-100. The second class has a small range of values composing it and it is possible that it makes it difficult for the neural network to understand the patterns. This issue can be attributed to the composition of the dataset mainly because is common to all the neural networks, even when it is not so evident, and not specific of a particular one: when not sure, the neural network chooses the first or the third class, but not the second.

An interesting point to observe is the ability of the neural network of recognising, even partially, the pattern characterising such a rare class as the fourth one, composed by only 2 samples. Even if it is not able in all the experiments to classify them as part of that, the neural network always makes the mistake of attributing the samples to the nearest class, the third one. This is really important since it means that the model learnt the pattern of the last and rarest classes, otherwise it would attribute the samples not recognised to the first and most populated class.

Due to the performances of the CNN, this is the neural network we decided to use for the last part of our experiments in which we study how much further in time we can forecast the class of DST to which the events belong.

CNN improvements

To try to improve the architecture chosen, starting from the neural network we improved the depth of the neural network or also tried to tune its hyperparameters such as the number of filters to use in each layer of the CNN itself.

The first test was done increasing the number of Convolutional layers to four for each block, where each original block can be thought as composed of two Convolutional layers and one Max Pooling layer. Adding the number of layers to the CNN can help the neural network in extracting features from the dataset passed as input thanks to the increased depth. This lets each layer and further blocks to extrapolate ore characteristics from the dataset to be analysed by the following layers. In table 5.21 the results are shown

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
Binary	98%	0.44	0.58	0.36
Multiclass	91%	0.40	/	/

Table 5.21: CNN results - 4 convolutional layers

After the increase of layers per block, we decided to add even more depth to the neural network adding more filters, but only to the first layer, leaving the rest of the neural network with the same number of filters as the architecture used to compute the baseline. Since usually when increasing the number of filters, we give the neural network the ability in learning more features from the dataset, adding more filters to the first block we give the chance to learn new features directly from the input data, before passing through all the layers of the neural network. In this way we also let the CNN extrapolate more characteristics from the dataset to be analysed by the following layers.

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
Binary	98%	0.46	0.44	0.48
Multiclass	91%	0.39	/	/

Table 5.22: CNN results -
4 convolutional layers and 16 filter

Despite the efforts, the tables 5.21 and 5.22 display that this change does not seem to improve the quality of our predictions, on the contrary, the value of the F1 score decrease or remained the same, even though the complexity of the neural network was higher. Even though in table 5.22 there is an improvement in the Recall value, it is not a huge one and for this reason, we decided to stick with the initial neural network, since complicating the architecture without any visible improvement is not helpful and may lead to overfitting or instabilities in the training process.

5.2.5 Forecasting hours

Our first intent is to forecast the important values of the DST on Earth for the reason we discussed in chapter 2. The importance of forecasting is linked to our ability to anticipate the events occurring to take precautions and avoid huge damages. Thus, it is important to not only forecast the current value, considering the past hours, but also to use these observations to predict future values. In this section we will evaluate how much in the future we can predict the classes of the events preserving a good quality in our results, considering as a starting point the cross correlation study we have done in chapter 3. For this purpose, we shifted the DST labels some steps in the future to let the neural network learn how to forecast, in particular, we chose to forecast in the future for 1, 2, 3 and 4 hours which were the most important hours in the cross correlation analysis.

Even in this case, we based our choices on both binary and multiclass classification.

For what concerns binary classification the results in table 5.23 may seem contradictory, the classifications made at 1 hour in the future have better results for what concerns the F1 score and also the Precision an Recall, which seem balanced, but on the other hand the results coming from the 4 hours forecasting reach our goal of having a higher Recall with respect the Precision and also the F1 score.

Window	**Accuracy**	**F1 score**	**Precision**	**Recall**
1 hour	99%	0.46	0.59	0.40
2 hours	91%	0.35	0.28	0.43
3 hours	98%	0.35	0.44	0.29
4 hours	94%	0.28	0.18	0.65

Table 5.23: Model forecasting - binary

The confusion matrix 5.24 regarding the forecasting at 4 hours shows the increase in the quality of the predictions, the number of true positive is higher than our baseline. This kind of outcomes let us forecast 4 hours in advance the events, with a higher accuracy, even though we have to deal with the high number of false positives.

	Predicted	
Actual	8188	432
	51	95

Table 5.24: CNN - binary confusion matrix

Considering the multiclass classification, the results are different from the ones

showed in case of binary one. Here the most accurate predictions are the ones interested in predicting the classes 2 hours in the future. The value of F1 score is distinguished by the other test, which has all the same results, both for accuracy and F1 score. In particular in the case of 4 hours forecasting the bias towards the first class is so high that even the events belonging to the fourth one are assigned to it.

Window	**Accuracy**	**F1 score**
1 hour	91%	0.34
2 hours	91%	0.41
3 hours	91%	0.34
4 hours	91%	0.34

Table 5.25: Model forecasting - multiclass

5.2.6 Training time

In table 5.26 the training time for each neural network model is reported being a metric influencing actively the choice of which neural network to use. Indeed, when a model is deployed and used not only for research purpose but also to recognise and forecast the events of interest, it is really important to have a response in a reasonable time. Moreover, when the model requires a huge amount of time in training, a less amount of time is available to do more experiments and investigate deeper the solutions applicable to that problem. We have also to think about the contribution of this time to the learning of the neural network itself when it does not allow the model to add knowledge it is a waste of time that could be used to train a better or improved model.

Model	**Binary**	**Multiclass**
LSTM	40’	60’
IndRNN	30’	50’
TCN	51”	52”
CNN	30”	40”

Table 5.26: Model training time per epoch

The table 5.26 shows a big difference from the CNN and TCN time required from training and the rest of the neural networks, this is the reason why many researches currently are focused on this kind of neural networks and in particular on the advantages of using them over the classical approaches conceived expressly for that purpose. This is mostly given by the ability of CNNs to learn, if not better,

surely in a smaller amount of time the pattern characterising a dataset due to their design and their ability to run in parallel.

Chapter 6

Conclusions and future works

6.1 Objectives and findings

This thesis final goal was to find a neural network able to predict the classes to which the DST belongs using the data collected about the features of the solar winds that from the Sun reach our planet. The complexity of this physical phenomena is reflected by the features we used and the difficulty in finding the best approach to correctly define the feature's values, while on the other hand, the relevant presence of missing values pushed us in searching for a method to address them, but only after considering all the characteristics of the features and a careful evaluation.

The proposed neural networks are simple, but effective in this ambit, although the quality of the dataset is not the best. The classical approaches performed better than the new architectures, despite their defects, with their ability in feature extraction and to learn a more complex pattern.

Forecasting in advance the events is important to protect both humans in orbit and instruments. This is the main objective of this thesis, and for this reason, we tried to train the neural network to learn to predict the class in the future, varying the gap between the observed features and the DST value.

What was evident in each experiment was the difficulty in letting the neural network divide the classes without leaning on one rather than the other. In all the tests the changes applied to the best model we selected led us to predictions giving more importance to one class with respect to the other, but not in an improvement of the forecasts and the correct identification of the classes. This could also be due to the pattern of the initial data, they are more concentrated in a definite interval while some and rare events, the most notable, depart from them. Being the data all concentrated in the same interval, the neural networks also have difficulties in understanding to which class the samples belong when they are too close to the

threshold values as in figure 6.1. In this figure, the green dots are the correctly classified events, while the red ones are the wrong ones and it is evident that the neural network struggles in this points.

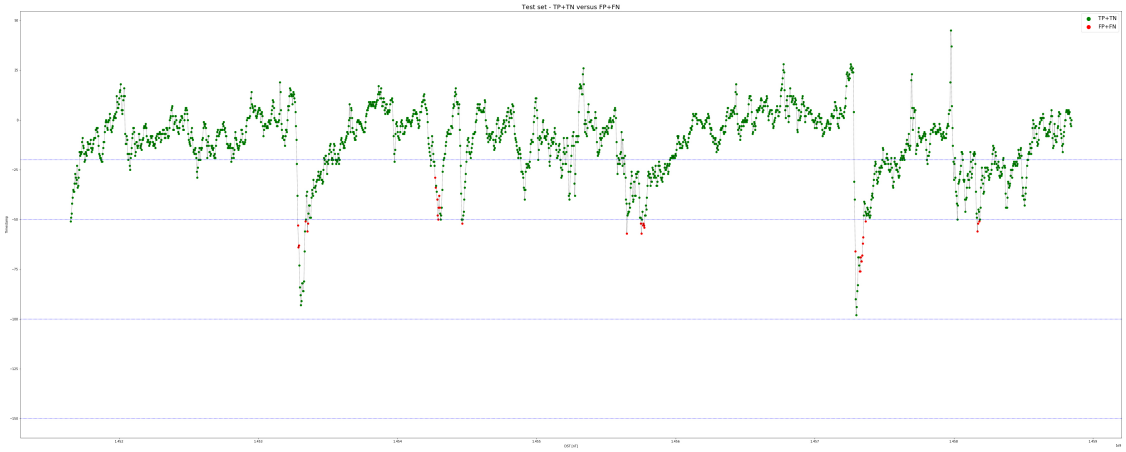


Figure 6.1: TP and TN versus FN and FP

6.2 Future work

For what concerns the possible future implementations and developments of this project we can think about two different ambits on which act: the dataset and the neural network involved.

6.2.1 Dataset

From the point of view of the dataset, we can act in different ways, starting from especially from the studies we have don in chapter 3. In the first moment, we could think to remove the seasonality from the features to understand whether new and valid information is hidden under it, together with possible trends. We could also think to address the problem of missing data using a different approach and trying to take advantage of the capacity of models in learning patterns to impute the missing samples.

A completely different and interesting point could be the use of a different representation of the dataset, not using anymore the time domain, but the frequency domain to represent the features. This change of representation could benefit the classification process giving to the neural network to understand better the pattern within.

6.2.2 Neural Network Model

The simplest improvement could be the usage of attention to the architectures used to let them understand which features are the most important and which are the time steps to take into account in this classification problem. Attention is now one the most used techniques in machine learning, especially in Natural Machine Translation (NMT), where to understand how to properly translate a sentence from one language to the other it is useful to understand which are the most important pieces of a sentence and where to put the translated words accordingly to the different grammars. In applying this method, we let the neural network decide which part of the sentence to maintain at each step and which not, this can also be applied to our problem to let the model understand which time steps are important in forecasting the class of the events.

An interesting ambit which may be worth to explore is the union of the past researches with the one conducted, especially link together the works on the Solar Dynamics Observatory's Helioseismic and Magnetic Imager instrument, based on images, with the work conducted in this thesis composing a bigger and more complex neural network able to join the patterns learnt from such different datasets. The complexity of such a neural network can increase the ability in forecasting and completeness in the description of such complex phenomena can be achieved. Following the idea of combining different datasets and purposes, we could train the same neural network both for classification and regression, even though our first purpose is to classify the events. In this way, the neural network could learn different patterns and improve its ability in predictions only by learning to address both tasks instead of the only classification.

To conclude, in such a vast an unexplored ambit such as the impact the solar activity has on the Earth, many more researches can be conducted, even using different datasets. The future investigations can take advantage of the many open points regarding this project and continue to improve our ability in forecasting such essential phenomena.

Bibliography

Bobra, M. G. and S. Ilonidis (2016), "Predicting Coronal Mass Ejections Using Machine Learning Methods", *The Astrophysical Journal*, 821, 2, p. 127, http://stacks.iop.org/0004-637X/821/i=2/a=127.

Bobra, Monica G and Sebastien Couvidat (2015), "Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm", *The Astrophysical Journal*, 798, 2, p. 135.

Chollet, F. (2017), *Deep Learning with Python*, Manning Publications Company, isbn: 9781617294433, https://books.google.it/books?id=Yo3CAQAACAAJ.

Florios, Kostas, Ioannis Kontogiannis, Sung-Hong Park, Jordan A Guerra, Federico Benvenuto, D Shaun Bloomfield, and Manolis K Georgoulis (2018), "Forecasting Solar Flares Using Magnetogram-based Predictors and Machine Learning", *Solar Physics*, 293, 2, p. 28.

He, Haibo and Edwardo A. Garcia (2009), "Learning from Imbalanced Data", *IEEE Trans. on Knowl. and Data Eng.* 21, 9 (Sept. 2009), pp. 1263-1284, issn: 1041-4347, doi: 10.1109/TKDE.2008.239, https://doi.org/10.1109/TKDE.2008.239.

Hochreiter, Sepp and Jürgen Schmidhuber (1997), "Long Short-Term Memory", *Neural Comput.* 9, 8 (Nov. 1997), pp. 1735-1780, issn: 0899-7667, doi: 10.1162/neco.1997.9.8.1735, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hunter, J. D. (2007), "Matplotlib: A 2D graphics environment", *Computing In Science & Engineering*, 9, 3, pp. 90-95, doi: 10.1109/MCSE.2007.55.

IBM Analytics (2018), *IBM SPSS Statistics*, [Online; accessed 31-August-2018], https://www.ibm.com/analytics/it/it/technology/spss//.

Lachin, John M (2016), "Fallacies of last observation carried forward analyses." *Clinical trials*, 13 2, pp. 161-8.

Li, Shuai, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao (2018), "Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN", *CoRR*, abs/1803.04831.

Little, Roderick J. A. (1988), "A Test of Missing Completely at Random for Multivariate Data with Missing Values", *Journal of the American Statistical Association*, 83, 404, pp. 1198-1202, doi: 10.1080/01621459.1988.10478722,

eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1988.10478722, https://www.tandfonline.com/doi/abs/10.1080/01621459.1988.10478722.

McKinney, Wes (2010), "Data Structures for Statistical Computing in Python", in *Proceedings of the 9th Python in Science Conference*, ed. by Stéfan van der Walt and Jarrod Millman, pp. 51-56.

Narasimhan, H., W. Pan, P. Kar, P. Protopapas, and H. G. Ramaswamy (2016), "Optimizing the Multiclass F-Measure via Biconcave Programming", in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1101-1106, doi: 10.1109/ICDM.2016.0143.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, 12, pp. 2825-2830.

Schafer, Joseph L. and John W. Graham (2002), "Missing data: our view of the state of the art." *Psychological methods*, 7 2, pp. 147-77.

Van der Walt, Stéfan, S. Chris Colbert, and Gaël Varoquaux (2011), "The NumPy Array: A Structure for Efficient Numerical Computation", *Computing in Science & Engineering*, 13, 2, pp. 22-30, doi: 10.1109/MCSE.2011.37, eprint: http://aip.scitation.org/doi/pdf/10.1109/MCSE.2011.37, http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37.