Optimising Seasonal Pasture Growth Predictions in Australia's NRM1010 Region: A Machine Learning Approach Utilising Pasture Field Data, Paddock Metadata, And Climate Data

Sevin Nejadi (N11131802)
IFN704 Assessment 3

1. Executive Summary

1.1 Objectives

This research project focuses on developing a robust, data-driven model to predict seasonal pasture growth in Australia's semi-arid NRM1010 region. The key objectives are to (1) identify the most influential environmental and management factors affecting pasture yield, (2) implement machine learning models, particularly regression and time-series forecasting, to achieve accurate and actionable predictions, and (3) evaluate model performance across different seasons and paddock characteristics to inform sustainable pasture management.

1.2 Methodology

To meet these objectives, a multi-stage research methodology was employed. First, datasets including pasture field data, paddock metadata, and climate information were sourced and pre-processed. Exploratory Data Analysis (EDA) helped in understanding temporal patterns, while dimensionality reduction and feature engineering were conducted to optimise the models. A Random Forest Regressor and Support Vector Regressor were tested for regression-based TSDM predictions, and two LSTM models were designed for time-series forecasting. Imbalance handling techniques were applied where necessary to improve model accuracy, and annual and seasonal data splitting ensured model evaluation aligned with seasonal patterns.

1.3 Key Findings

The study highlights key variables influencing TSDM predictions and compares the performance of various models. Paddock size emerged as the most significant factor, followed by solar radiation, relative humidity at minimum temperature, and rainfall, all of which are crucial for productivity, especially in semi-arid environments. Seasonal patterns were also important, with spring identified as a peak growth period. In model performance, SVR achieved the highest accuracy (85.73%) with a MAPE of 0.143 but required a longer processing time. The simpler Forecasting LSTM model provided balanced performance, with an accuracy of 84.11%, a MAPE of 0.159, and efficient processing time, making it a viable option for practical use. This comparison underscores the importance of model selection for agricultural forecasting, as both effectiveness and computational efficiency are critical for applications in pasture management.

1.4 Reflections

The project provided valuable insights into forecasting Total Standing Dry Matter (TSDM) in agriculture, emphasizing the significance of data quality and preprocessing for reliable predictive models. Integrating paddock metadata, climate data, and TSDM data highlighted the necessity of thorough data cleaning, especially when faced with challenges like imbalanced data distributions across paddocks and seasons. Techniques such as oversampling were essential to improve model generalisation. The iterative process of model selection underscored the importance of balancing complexity with interpretability, revealing that simpler models could achieve comparable accuracy with lower computational demands. Collaboration and mentorship further enriched the learning experience, enhancing my understanding of machine learning applications in agriculture and the factors affecting model performance in practical scenarios.

1.5 Future Work

Future work in this area should focus on several key directions to enhance TSDM prediction models. Integrating additional data sources, such as remote sensing imagery from satellites and drones like Landsat and MODIS, could provide more detailed insights into pasture conditions and growth dynamics, potentially improving model accuracy. The application of hybrid modeling techniques, including ensemble methods that leverage predictions from multiple algorithms, could further enhance performance and reliability. Additionally, optimizing hyperparameters and conducting extensive cross-validation will be crucial for ensuring model robustness across diverse datasets and environmental conditions. Addressing imbalanced data through advanced resampling methods, cost-sensitive learning, or synthetic data generation is also vital for improving predictive accuracy. Finally, incorporating the impacts of climate change on pasture growth dynamics into predictive models will support stakeholders in making informed decisions regarding sustainable pasture management.

1.6 Conclusion

This study successfully demonstrates the potential of machine learning models for seasonal pasture growth prediction in semi-arid regions. The findings highlight the influential role of climate variables and underscore the need for season-specific models to optimise accuracy. By advancing data-driven approaches in pasture management, this research supports sustainable agricultural practices that can adapt to the growing challenges of climate variability.

2. Introduction

The effective management of pasture growth is essential for optimising farm productivity and sustainability. Pastures serve as a primary food source for livestock, directly influencing the economic livability of farming enterprises. However, pasture growth is inherently dependent to a multitude of factors including climatic conditions, soil fertility, water and management practices. Moreover, with climate change and its effects and variability on putting more pressure on farming, the need for accurate forecasting models that can predict pasture growth has become more urgent than ever. Leveraging these models assist farmers in making optimised decisions about stocking rates, supplementary feeding, and paddock rotation, maintaining sustainable supply chain.
Recent advances in remote sensing technology, especially with Sentinel-2 satellite imagery, have opened up new opportunities with the possibility for closely monitoring pasture conditions. With the ability to capture detailed images over time providing Pasture yield data, this technology, when combined with paddock metadata information and climate data, provides the opportunity to implement highly accurate models for predicting pasture growth. However, utilising all these different types of data together presents several challenges. There's the need to handle large

dataset, the complexity of understanding how different factors interact, and the need to develop models that work well both in general and in specific local environments.
In the context of the NRM1010 region of Australia, a semi-arid zone characterised by highly variable rainfall and soil types, accurate pasture growth forecasting is particularly critical. This region poses unique challenges due to its environmental variability and the specific needs of the agricultural sector operating within it [15]. Therefore, developing a forecasting model that can cater to the specific conditions of this region is not only beneficial for local farmers but also contributes to the broader body of knowledge on pasture management in semi-arid environments.

3. Literature Review

The study of pasture growth prediction has developed significantly over the years, with early models primarily relying on simple empirical relationships between environmental variables and pasture biomass. These models often used straightforward statistical approaches to correlate variables like temperature, precipitation, and soil moisture with pasture growth. However, they faced significant limitations in capturing the complex interactions between multiple factors influencing pasture biomass. For instance, early models struggled with non-linear relationships and interactions among environmental variables, leading to oversimplified predictions that did not account for the variability in pasture growth due to diverse climatic conditions or soil types. Additionally, these models often relied on limited available data sources, which restricted their ability to generalise across different regions and conditions. Due to the limitations, the accuracy and reliability of these models were not as strong as they required to be. This has led to the development of more advanced methods and approaches, like using machine learning, to better manage the complexity and richness of the available data [1], [2].

Overview of Satellite Imagery in Agricultural Forecasting

Due to the ability of providing comprehensive and frequent data over large-sized areas, satellite imagery has become an essential tool in agricultural forecasting. Sentinel-2 satellites, via multispectral sensors, deliver high-resolution imagery that captures vegetation health dynamics, which is critical for accurate pasture growth prediction [3], [4]. These images present insights into vegetation indices, such as the Normalised Difference Vegetation Index (NDVI), which are broadly used to assess pasture health and biomass [5], [6].
The application of Sentinel-2 data has been explored in various studies for monitoring pasture conditions and growth patterns. For instance, Brown et al. [1] highlights the effectiveness of NDVI in estimating biomass and predicting pasture yield in diverse climatic conditions in Northern Australia [1]. Similarly, Stumpe et al. [8] demonstrates the potential of combining Sentinel-2 imagery with climate data to enhance the accuracy of pasture growth forecasts [8]. The integration of satellite data conducted over the southern Belgium with ground-based observations can further improve model performance, as shown in [12].

Machine Learning Models in Predictive Agriculture

Machine learning models have shown significant promise in agricultural forecasting by enabling the extraction of complex patterns from large datasets. These models can handle diverse data sources, including satellite imagery and climate data, to predict various agricultural outcomes [7], [8]. For instance, Barrett et al. [2] discusses the use of machine learning algorithms to predict crop yield in the temperate regions of northwest Europe based on remote sensing data, demonstrating their potential for similar applications in pasture growth forecasting [2].
Several studies have explored the application of different machine learning techniques, such as decision trees, support vector machines, Random Forest and deep learning models, for predicting pasture growth. Correa-Luna et al. [9] illustrates the use of ensemble methods to combine multiple predictive models, enhancing the accuracy of pasture yield forecasts across 14 farms in New South Wales, Australia [9]. Additionally, Nickmilder et al. [10] emphasise the importance of feature

selection and extraction in improving model performance in a study conducted in the Walloon region of Belgium, highlighting its crucial role in identifying significant factors influencing pasture yield [10].

Climate Data and Its Influence on Pasture Growth

Climate data plays an important role in understanding and forecasting pasture growth, as it influences various aspects of vegetation development. Factors such as temperature, precipitation, and soil moisture directly impact pasture productivity [11], [12]. Morse-McNabb et al. [11] explores the relationship between climate variables and pasture growth in southeast Australia, highlighting the importance of integrating climate data into forecasting models [11]. The study also discusses different approaches for incorporating climate data, such as using historical climate records and real-time weather information, to enhance model accuracy.

Defining the Project Scope and Methods

The scope of this study involved developing a forecasting model for pasture growth based on machine learning techniques in the NRM1010 region of Australia, utilising NRM1010 TSDM data which is a pre-processed data from Sentinel-2 satellite imagery, Paddock metadata and climate data. The analysis focused on seasonal and annual pasture growth patterns, providing insights into how these temporal factors impact pasture dynamics.

The main focus of the project was on identifying significant features influencing pasture yield and selecting the most relevant variables for predictive modelling. The methodology contained data pre-processing, exploratory data analysis, feature extraction, feature selection, and model training using various machine learning algorithms etc. The details of proposed machine learning model are provided in section 5. Proposed Methods.

The literature review indicates that a combination of pasture field and climate data, as well as advanced machine learning techniques, show significant improvement on pasture growth forecasts. By using these tools, the project aimed to provide valuable insights for optimised farm management and decision-making in the NRM1010 region.

4.Materials

This project aimed to develop a comprehensive predictive model for pasture growth in the NRM1010 region of Australia, leveraging the integration of Pasture field data, paddock metadata, and SILO climate data. Below the details of the domain datasets followed by data analysis are provided.

4.1. Datasets

The project began with an extensive data exploration phase, focusing on cleaning and preprocessing the proposed domain datasets.

NRM1010 TSDM data

The NRM1010 TSDM data, pasture field data, provides critical information on pasture yield, measured as total standing dry matter (TSDM) in kilograms of dry matter per hectare (kg DM/ha). Collected every 15 days from 2017 to 2023 using Sentinel-2 satellite imagery across multiple paddocks in Australia's NRM1010 region. This dataset has been pre-processed to remove noise using a 30-day median smoothing technique and includes necessary corrections for cloud coverage, tree coverage, and fence effects.

Paddock metadata

In addition to the TSDM data, paddock metadata proposed to incorporate information about crop types (pasture species) and paddock sizes collected from 2017 to 2023. Each paddock may host different pasture species, such as native grasses, lucerne, or rye. This metadata is significant for understanding the diverse pasture species in the region, and their potential impact on pasture yield.

SILO climate data

The SILO climate data includes climatic factors collected from SILO, which uses interpolation techniques based on Bureau of Meteorology data to create spatial grids. This dataset provides daily records of rainfall (mm), temperature (°C), humidity (%), evaporation (mm), and radiation (MJ/m2) from 2017 to 2022.

4.2. Data Analysis

In Figure 1, the TSDM mean shows an overall gradual decline initially, followed by periods of stabilization and slight increase towards the end. The decline in TSDM in the earlier years could be related to lower rainfall or other environmental stressors, while the stabilization in recent years may indicate better management practices or more favourable conditions.

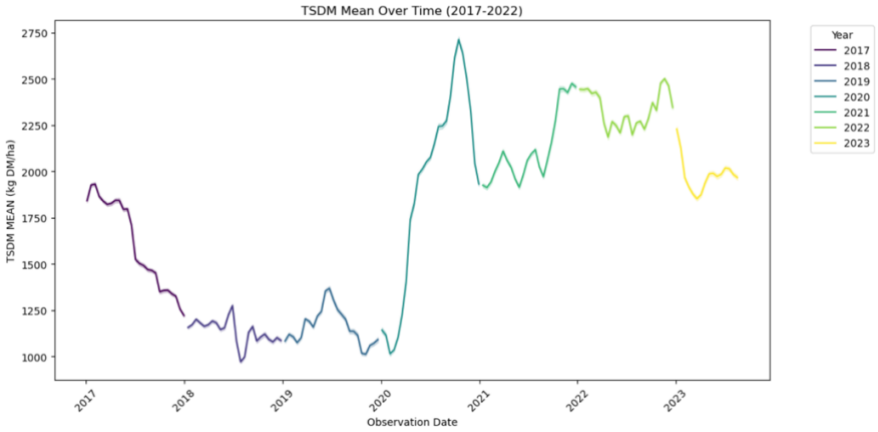

Figure 1- Line plot of TSDM mean over time by Year.

5. Proposed Methods

As mentioned earlier, the aim of this study is to produce reliable TSDM predictions and enhance understanding of the key drivers of pasture productivity in the NRM1010 region.

5.1. Data Exploration and Preprocessing

The first stage of the project involved exploring and preprocessing the available datasets to prepare them for analysis, focusing on loading, cleaning, feature engineering, data integration, and exploratory data analysis (EDA). This is crucial for ensuring that the data is clean, well-organised, and ready for feature extraction and modelling. Source code for data Exploration and preprocessing is available in Appendix A.

5.1.1. Data Loading

Data preprocessing started with loading the NRM1010 TSDM data, paddock metadata, and the SILO climate datasets using Pandas Python package.

The NRM1010 TSDM data consists of a total of 1,030,320 data records and three features. PADDOCK_ID a unique identifier for each paddock; OBSERVATION_DATE recorded every 15 days from 2017 to 2023, and TSDM_MEAN which indicates the amount of pasture in a paddock at a given

time measured as total standing dry matter (TSDM) in kilograms of dry matter per hectare (kg DM/ha).
The paddock metadata with 17,966 entries includes PADDOCK_ID a unique identifier for each paddock, CROP_TYPE, which indicates the pasture species; LANDSIZE_HA, which indicates the size of the paddock in hectares; and PASTURE_STATE, which records the type of pasture in the corresponding paddock with values such as Grazing, Cropping, Yard, Pen, Laneway, Vegetation, Feedlot, and Hay. There is also a CREATION_DATE variable, recorded daily from 2017 to 2023.
The climate data with 33,276,908 entries from 2017 to 2022 includes PADDOCK_ID, a unique identifier for each paddock; DATE, indicating the recording date; RAIN, which shows the rainfall amount in millimetres; and MAX_TEMP and MIN_TEMP, which indicate maximum and minimum temperatures (°C), respectively. RH_TMAX and RH_TMIN represent the percentage of relative humidity at maximum and minimum temperatures, while EVAP denotes the evaporation rate in millimetres. RADIATION indicates the level of solar radiation measured in megajoules per square meter (MJ/m²).

5.1.2. Data Preprocessing

Each of the datasets further refined by addressing any missing values, removing noise, and ensuring that all datasets (NRM1010 TSDM, paddock metadata, and SILO climate data) are properly aligned.

5.1.2.1. The following steps were implemented for NRM1010 TSDM data:

TSDM_MEAN identified with 19,055 instances with null values while there is no TSDM_MEAN with value equal to zero.

OBSERVATION_DATE converted to datetime format.

Feature Engineering for Yearly and Seasonal Patterns

Given the study's focus on analysing yearly and seasonal patterns of pasture growth, two new features were derived from the OBSERVATION_DATE variable:

1. **Year Feature:** The Year feature was extracted directly from the OBSERVATION_DATE variable, enabling annual trend analysis.
2. **Season Feature:** To capture seasonal variations, a custom function was defined to categorise each observation date into a corresponding season:

This function was applied to the month component of each observation date to create a Season column, where:

- **Summer** corresponds to December, January, and February
- **Autumn** to March, April, and May
- **Winter** to June, July, and August
- **Spring** to September, October, and November

By engineering these features, the dataset was structured to support an in-depth analysis of seasonal and yearly pasture growth patterns, enabling insights into how these temporal factors impact pasture dynamics.

Interpolation of TSDM_MEAN Variable

To preserve data integrity while allowing for accurate seasonal trend analysis, the following steps were implemented to address missing values in the TSDM_MEAN variable:

1. Initial Assessment: No instances of TSDM_MEAN had values equal to zero, ensuring that all zeros introduced subsequently would represent interpolated values.
2. Identification of Missing Values: The dataset comprises 6,360 unique paddock IDs within the NRM1010 region, of which 19,055 instances had null TSDM_MEAN values.

3. Filtering of Null-Only Paddocks: Eighteen unique paddock IDs contained null TSDM_MEAN values across all instances. A subset of 2,916 rows within the NRM1010 TSDM data corresponded to these paddock IDs.
4. Exclusion of Null-Only Paddocks: The 18 paddock IDs with all-null TSDM_MEAN values were not present in the climate or metadata datasets, allowing safe removal of their corresponding rows from NRM1010 TSDM data.
5. Replacement of Remaining Missing Values: The remaining NaN values in TSDM_MEAN were initially replaced with zero to facilitate subsequent interpolation.
6. Mean Calculation for Interpolation: The dataset was grouped by PADDOCK_ID and Season, calculating the mean TSDM_MEAN within each group.
7. Interpolation of Zeros: Zeros in TSDM_MEAN were replaced with the calculated seasonal mean for each paddock, ensuring consistency within each paddock's seasonal data.
8. Creation of Interpolated Column: A new column, TSDM_MEAN_Interpolate, was generated to store the interpolated values, reflecting the average TSDM_MEAN by PADDOCK_ID and Season.

Final Grouping for Seasonal and Yearly Analysis

To facilitate the study's focus on seasonal and annual patterns of pasture growth, the NRM1010 TSDM data was grouped by PADDOCK_ID, Year, and Season. Within these groups, the mean value of the interpolated TSDM_MEAN was calculated to serve as the final dataset, optimised for this study's objectives. This seasonal grouping method enables a detailed examination of how pasture growth varies across different seasons, providing valuable insights for both agricultural planning and research analysis. By structuring the data in this way, the study is equipped to explore temporal growth patterns and their implications for effective pasture management.

5.1.2.2. The following steps were implemented for Paddock metadata:

CREATION_DATE converted to datetime format.

The dataset was checked for missing values, and the CROP_TYPE variable was found to contain some null values. These missing values were addressed by replacing them with the string 'Unknown.'

Since grazing pastures are exclusively used for feeding livestock, only instances with a PASTURE_STATE of 'Grazing' were retained.

5.1.2.3. The following steps were implemented for Climate data:

Firstly, DATE converted to datetime format.

Feature Engineering for Yearly and Seasonal Patterns

Given the study's focus on analysing yearly and seasonal patterns of pasture growth, two new features, Year and Season, were derived from the DATE variable, similar to the approach used with the NRM1010 data.

The climate data contained no missing values. However, upon further examination, it was found that the EVAP (evaporation) variable had some zero values. While evaporation rates can approach zero at 0°C, it is uncommon for them to be exactly zero unless all contributing factors (such as wind, humidity, and radiation) also strongly limit the process. Additionally, the exploration showed instances of EVAP = 0 even when the minimum temperature was above zero. To investigate further, the rain condition was examined to see if EVAP = 0 might have been recorded on rainy days. The results showed that out of 103,475 rows with EVAP = 0, 88,290 rows also had rain. However, there were still 15,185 rows with EVAP = 0 without any rainfall. Thus, we cannot conclude that EVAP = 0 was recorded exclusively when it was raining.

Interpolation Technique for the EVAP Variable

To handle zero values in the EVAP variable, an interpolation approach was applied based on the mean values for each PADDOCK_ID, Year, and Season. The steps taken were as follows:

1. Grouping and Calculation of Mean EVAP: The data was grouped by PADDOCK_ID, Year, and Season to calculate the mean EVAP for each group, allowing for more context-specific interpolation values.
2. Replacing Zero Values: The zero values in EVAP were then replaced with the calculated mean for the respective PADDOCK_ID, Year, and Season combination.
3. Creating a New Column: A new column, EVAP_Interpolate, was created to store the interpolated EVAP values, ensuring that the original data remained unchanged.

Final Grouping for Seasonal and Yearly Analysis

Similar to the approach used with the NRM1010 TSDM data, and with a focus on seasonal and annual patterns of pasture growth, the Climate data also was grouped by PADDOCK_ID, Year, and Season.

5.1.3. Data Integration

Firstly, all PADDOCK_ID values in the three datasets were converted to lowercase strings to ensure proper alignment across datasets. Subsequently, the three pre-processed datasets were integrated by connecting records based on PADDOCK_ID, Year, and Season. The final merged dataset was then checked for duplicate instances and null values, and the index was reset.

5.1.4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis was conducted to investigate datasets and visualise the main characteristics, patterns and temporal trends of the data.

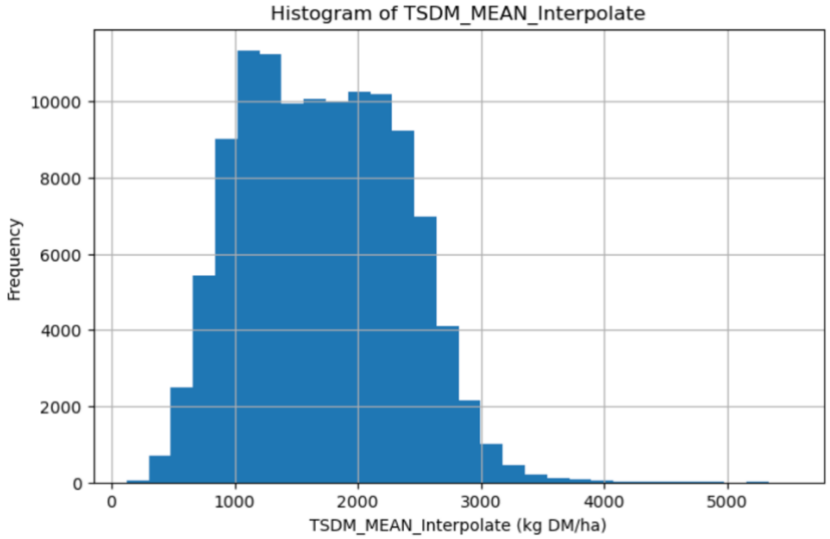

Figure 2- A Histogram of TSDM mean

The distribution of TSDM mean values has a longer tail on the right side. This indicates that most paddocks have relatively low to moderate values of TSDM mean, while a smaller number of paddocks have much higher values, potentially due to factors like better soil quality, water availability, or favourable climate conditions.

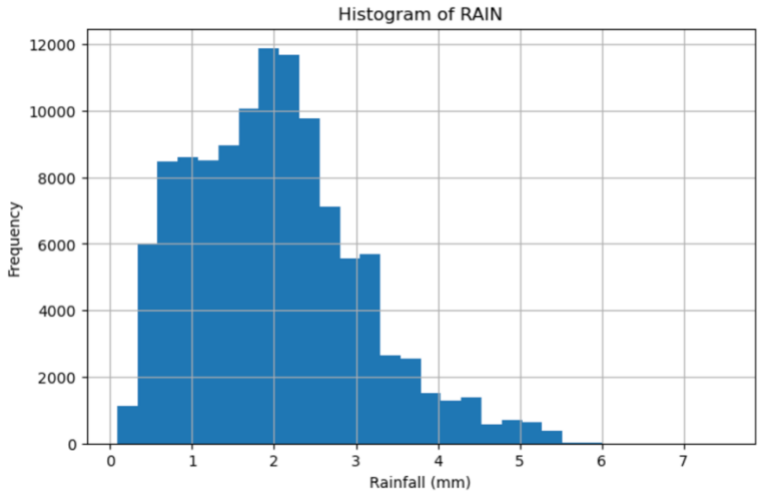

Figure 3- A Histogram of rainfall

The distribution of rainfall pattern suggests that light to moderate rainfall is common, whereas heavier rain events are rare. Such a distribution is typical in semi-arid regions where light, frequent rainfall occurs, but there are occasional heavy rainfall events, which can have important implications for water availability, soil moisture, and pasture growth.

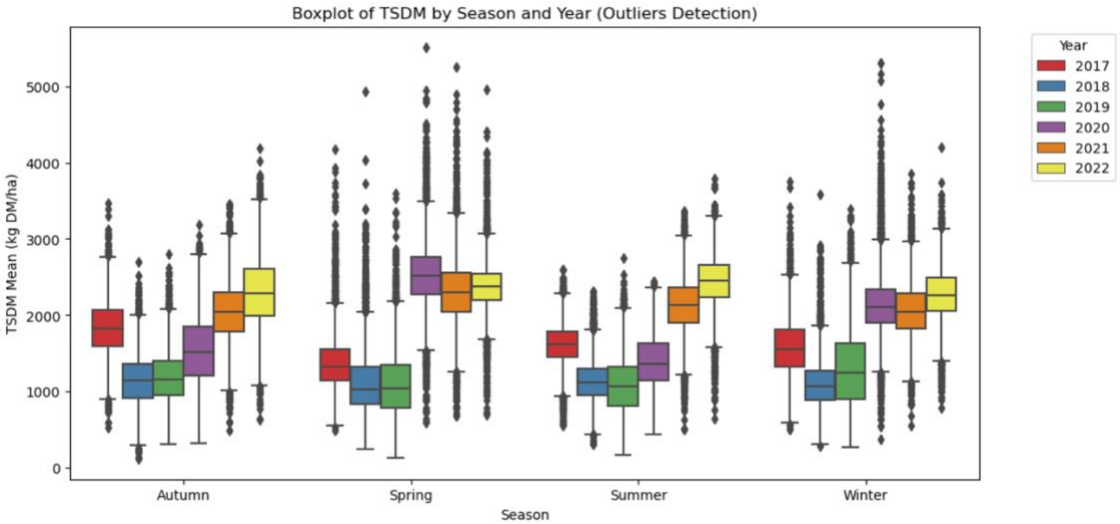

Figure 4- The boxplot of the distribution of TSDM across different seasons for each year from 2017 to 2022. Each colour represents a different year, and the plot highlights the variation in TSDM values within each season and across years, with outliers marked as points outside the whiskers.

This boxplot as shown in Figure 4, indicates clear seasonality in TSDM, with spring showing the highest productivity on average by having generally higher TSDM values compared to other seasons. The variations in TSDM across years highlight the influence of annual climate conditions or management changes on pasture growth. Additionally, the presence of many outliers suggests variability in pasture growth, likely due to variations in specific conditions such as extreme weather events, soil health, or management practices within specific paddocks.

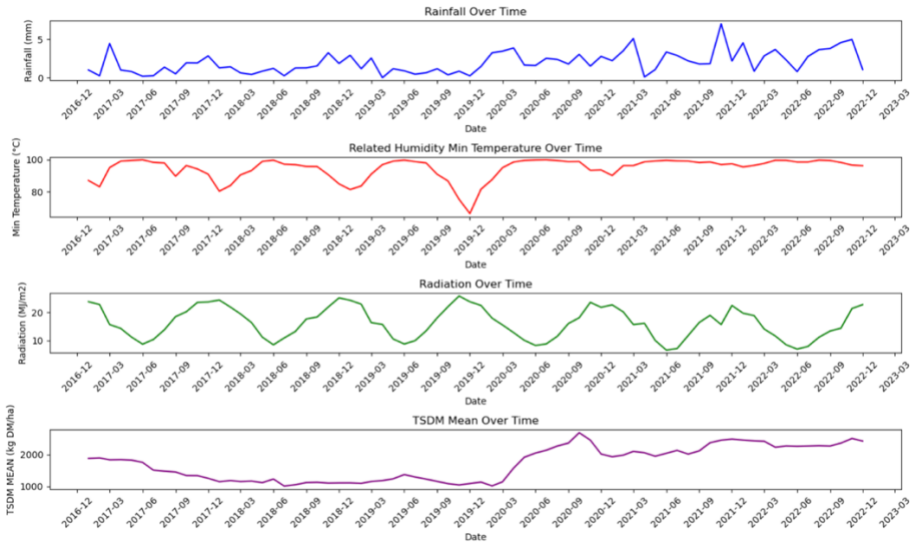

Figure 5- Multi-line plot of Climate Factors (Rainfall, Relative humidity at minimum temperature, Radiation) and TSDM Over Time.

This plot collectively highlights the relationship between climate factors and pasture growth (TSDM). The fluctuations in rainfall and relative humidity at minimum temperature likely contribute to the variability observed in TSDM. Solar radiation, being relatively stable and cyclical, aligns with seasonal growth patterns but does not show a direct impact on the long-term trends of TSDM. The TSDM trend suggests that water availability (rainfall and humidity) might be more influential in determining pasture growth than radiation alone.

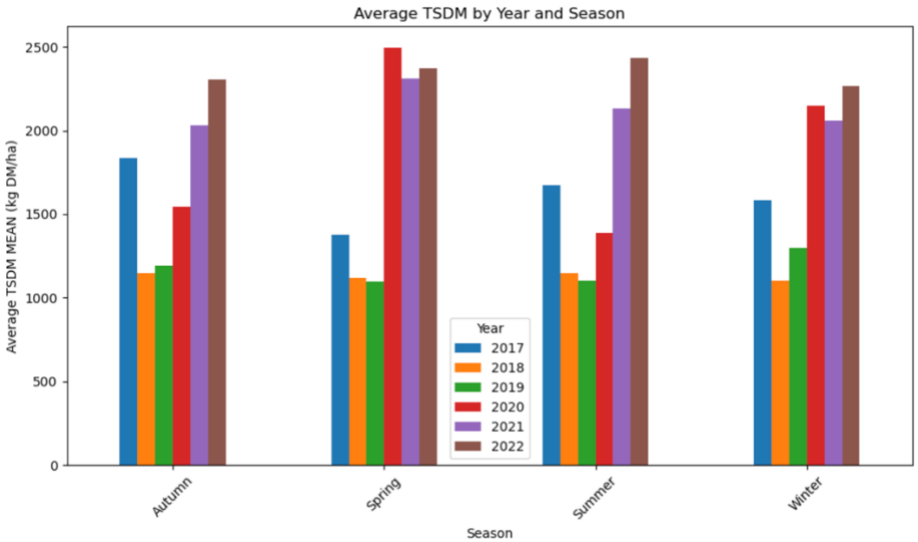

Figure 6- The bar plot of the average TSDM Mean for each season across different years.

This bar plot shows TSDM values are generally higher in spring and summer for most years, suggesting that pasture growth peaks during warmer seasons. In general year 2022, shows higher TSDM across all seasons indicating that this year experienced better overall conditions probably due to better management practices, adequate rainfall, or favourable temperatures.

These analysis supports the importance of considering multiple environmental factors, especially water-related ones, when predicting pasture productivity. Moreover, this information can help to identify periods and conditions that lead to higher productivity of pasture improving pasture management practices.

5.1.5. Feature Selection

A correlation matrix for continuous variables using the Pearson correlation used to examine the relationships between pasture yield (TSDM) and predictors. The aim was to Identify the significant features that influence pasture yield (TSDM) and select the most relevant ones for predictive modellings. The source code for Feature selection is available in Appendix B.

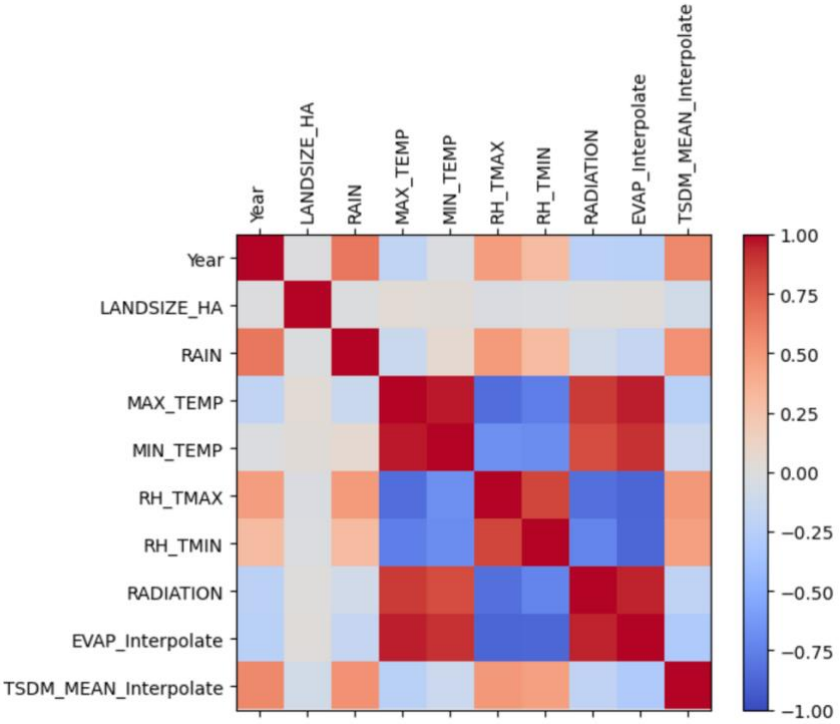

Figure 7- A correlation matrix was used to understand the relationships between features, and to detect highly correlated features.

As shown in Figure 7, evaporation is highly correlated with both maximum and minimum temperature and radiation. Minimum and maximum temperatures are also highly correlated, among other relationships. maximum temperature and radiation are also highly correlated. Additionally, there is a strong negative correlation between evaporation and relative humidity for both maximum and minimum temperatures. Furthermore, relative humidity for maximum temperature is highly negatively correlated with both maximum temperature and radiation.

The predictors Year and rainfall have a noticeable but not strong influence on the Total Standing Dry Matter (TSDM). The relationship with Year may suggest trends over time, while the relationship with rainfall may indicate how moisture availability impacts pasture growth. Understanding these relationships can provide valuable insights into how TSDM varies with seasonal and yearly changes. Since the relationship is moderate, it indicates that while Year and rainfall are useful for predicting TSDM, they may not fully explain its variability, and there are also other factors affecting TSDM that are not captured by these predictors.

5.1.6. Dimensionality Reduction

Based on the outcome of the correlation matrix, dimensionality reduction was conducted. The features 'Year', 'Season', 'CROP_TYPE', 'LANDSIZE_HA', 'RAIN', 'RH_TMIN', and 'RADIATION' were selected to remain as independent variables, while the features 'MAX_TEMP', 'MIN_TEMP', 'RH_TMAX', and 'EVAP_Interpolate' were excluded. 'TSDM_MEAN_Interpolate' was selected as the target feature.

5.1.6. Data Transformation: One-Hot Encoding and Standardization

One-Hot Encoding applied to convert categorical features Season and CROP_TYPE into binary columns. Z-score Standardisation method used to transform the scale of the numerical features to have a mean of 0 and a standard deviation of 1, which helps many machine learning models perform better.

5.1.7. Temporal Data Splitting: Train, Validation, and Test Sets

After applying the necessary transformations, the merged dataset was split into training, validation, and test sets based on the 'Year' column. Specifically, the data was partitioned as follows: the training set included 76,784 instances from 2017 to 2020, the validation set consisted of 19,196 instances from 2021, and the test set comprised 19,196 instances from 2022. This temporal splitting approach ensured that the model was trained on past data and evaluated on future data, allowing for a more realistic assessment of its performance. The temporal approach in splitting the data enhances the model's ability to learn from past trends and be tested on future data, which is important for enhanced predictive modeling in real-world applications.

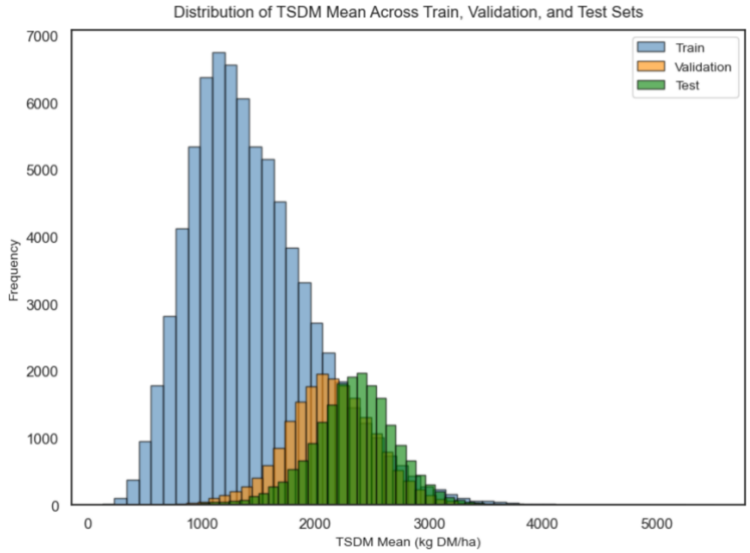

Figure 8- The distribution plot of Total Standing Dry Matter across Train, Validation, and Test sets.

Figure 8 shows that the training set distribution suggests that while the majority of training observations are low, there is a significant frequency of values at the higher end, indicating that the model will have exposure to a range of TSDM values. However, the significant concentration of lower values and right skew, indicates potential challenges in accurately predicting higher TSDM values. This might present challenges in predicting higher values later, and strategies to handle the imbalance effectively are required.

A normal distribution in the validation set implies that the model will face a consistent range of values that reflect what it might encounter in real-world scenarios. The similarity in the test set distribution indicates that the model's performance on the test set will be indicative of how well it generalises to similar conditions.

5.1.8. Imbalance Handling

To address imbalanced data in the training set before training the proposed models, oversampling of the minority class was considered to handle the limited availability of higher TSDM values. This method increases the number of samples in the minority class by duplicating them. Details of the oversampling technique used are provided in the Experimental Evaluation section.

5.2. Proposed Random Forest Regressor

Random Forest (RF) regression algorithm has been widely applied in agricultural and ecological research, particularly for predicting outcomes in pasture, vegetation, and crop growth, due to its robustness and adaptability with complex datasets as indicated in [9, 11, 12, 13, 14, 19]. Literature indicates that RF consistently delivers high performance in scenarios with high-dimensional input variables, non-linear relationships, and noise, making it a strong candidate for Total Standing Dry Matter (TSDM) prediction. RF also known for the ability of overfitting handling [16]. In this study, the RF Regressor was selected not only for its predictive accuracy but also for its feature importance capability, which is valuable for identifying the most influential factors impacting pasture growth [10]. By leveraging these strengths, this study aims to produce reliable TSDM predictions and enhance understanding of the key drivers of pasture productivity in the NRM1010 region.

A Random Forest regression method , optimised using Grid Search, was initially developed to predict TSDM, and to leverage feature importance.

5.3. Proposed Support Vector Regression

Similar to Random Forest, Support Vector Regression (SVR) has been widely applied in agricultural projects [16, 17, 18, 19, 20] and is well-suited for pasture yield prediction due to its robustness, adaptability, and efficiency with smaller datasets [16]. By focusing on a subset of data points, known as support vectors, SVR effectively handles noise and outliers—an advantage when working with TSDM data that may contain fluctuations or atypical values. With flexibility in kernel functions, such as linear or radial basis functions, SVR can capture complex, nonlinear relationships often found in TSDM datasets influenced by various environmental factors [16, 17]. Additionally, SVR performs reliably on smaller datasets, providing accurate predictions without requiring extensive data. Its capacity to control the prediction error margin through parameters like epsilon allows SVR to balance bias and variance, enhancing generalization and improving TSDM prediction accuracy. These characteristics make SVR an effective method for modeling the complexities of TSDM prediction.
A Support Vector regression method , optimised using Grid Search, was initially developed to predict TSDM.

5.4. Proposed Time-series Forecasting model using LSTM

Furthermore, to deepen the study, a time-series forecasting model using LSTM (Long Short-Term Memory) was implemented to capture the temporal dependencies in pasture growth. This model provided insights into how past patterns in climate and management practices influence future pasture yield.
Pasture growth is influenced by a sequence of past events, such as previous weather conditions, management practices, and seasonal patterns. These temporal dependencies mean that what happened in the past can significantly impact future growth. For instance, a period of drought can affect not only the immediate yield but also how the pasture recovers in the following months.
LSTM (Long Short-Term Memory) is a type of deep learning recurrent neural network (RNN) architecture designed to handle long-term dependencies in data, and is commonly used for sequential data analysis, including time series analysis and predicting [7, 8].
LSTM networks are suitable for sequential data modelling where the network learns to predict the next value or sequence of values in a sequence based on previous observations. In the context of pasture growth, LSTM can model the intricate interplay between various factors (e.g., delayed effects of past rainfall or temperature changes) over time.
Developing a time-series forecasting model using LSTM for predicting pasture growth can offer significant advantages. By capturing temporal dependencies, the model can accurately forecast long-

term trends, considering that pasture growth at any given time is influenced not only by current conditions but also by those in preceding days, weeks, or even months. Additionally, this approach enhances the understanding of seasonality. Since pasture growth often follows seasonal patterns, time-series models can identify and incorporate these effects, leading to more reliable predictions throughout the year.

5.5. Proposed Time-series Forecasting model using LSTM (Complex)

Lastly, to investigate potential performance improvements, a more complex time-series forecasting model using LSTM was implemented by adding more layers to the previous model.

5.6. Performance evaluation Metrics

To evaluate each model's performance, the MAPE and Accuracy (1 -MAPE) metrics were used.

- MAPE: Measures the average percentage error between predicted and actual values.
- Accuracy: Calculated as 1 - MAPE, which gives a percentage of how close the predictions are to the true values.

MAPE (Mean Absolute Percentage Error) is highly suitable for evaluating pasture growth prediction methods due to its focus on percentage-based error. MAPE provides an intuitive measure of prediction error by expressing it as a percentage, which helps to assess model performance relative to the scale of the data. This percentage-based metric is particularly helpful for agricultural data like TSDM (Total Standing Dry Matter) since it allows for a straightforward interpretation of error across various ranges of pasture yield values, making it easier to compare models and interpret results meaningfully in real-world terms.

Additionally, using the corresponding accuracy metric derived from MAPE (Accuracy = 1 - MAPE) is beneficial by translating the MAPE value into an easily understandable percentage that reflects the how close the model is to the actual values, giving stakeholders a clear indication of how well the model is performing in practical terms. Together, these metrics offer a balanced view of model performance, where MAPE highlights the prediction error, and Accuracy provides a straightforward measure of model effectiveness.

6. Experimental Evaluation and Results

Below the details of operating environment, followed by details for each of the proposed methods as well as performance evaluation measures used for the experimental evaluation are provided. Source code for proposed methods is available in Appendix C.

6.1. Operating Environment

The experiment was performed in the Anaconda Jupyter Lab 3.5.3 environment using Python 3.10.9 for implementation.

6.2. Random Forest Regressor Implementation

A Random Forest Regressor implemented to predict Total Standing Dry Matter (TSDM) values, utilising both a grid search for hyperparameter optimisation and a weighted approach to handle the

data's imbalance to improve balance across predictions. This setup allows the model to account for imbalanced data during grid search by applying sample weighting only to training data.

To tune the Random Forest model effectively, a parameter grid is created with various combinations of hyperparameters. These include n_estimators, which defines the number of trees in the forest, max_depth to control tree depth and prevent overfitting, min_samples_split and min_samples_leaf to determine minimum requirements for splitting and leaf nodes, and bootstrap to toggle whether samples are drawn with replacement. The grid search then evaluates combinations of these parameters to find the best fit, based on minimising mean squared error.

The training and validation data are merged into a single set, with predefined indices ensuring that the grid search respects the intended split. This setup allows the model to generalise well across diverse TSDM values, guided by optimal parameters identified by grid search. The best model is evaluated using Mean Absolute Percentage Error (MAPE) and Accuracy (1 - MAPE), allowing stakeholders to evaluate both the error margin and prediction accuracy across the validation and test sets.
Overall, the grid search is performed on train set, which include data from 2017 to 2020. After grid search completes, the best model is evaluated on validation set (data from 2021) to check its performance before finalising. finally, once the best parameters are found, the model's generalization performance is tested on test set (data from 2022). PredefinedSplit was used to define and explicitly guide GridSearchCV to split between the training and validation sets during the cross-validation process. This method can potentially lead to better accuracy and more robust hyperparameter tuning by guiding the grid search to use both training and validation data in a controlled way.

6.2.1. Results

The best-performing Random Forest Regressor model was selected through a grid search process. The Table 1summarises the optimal hyperparameters and the Table 2 demonstrates the model's performance metrics across training, validation, and test datasets.

The model achieved a **Mean Absolute Percentage Error (MAPE)** of 0.177 on the test dataset, corresponding to a **prediction accuracy of 82.25%**. The lower MAPE on the validation set (0.073) suggests good model performance during tuning, though the slight increase in MAPE on the test set indicates some variance in predictive accuracy across unseen data. These results demonstrate the model's effectiveness in predicting TSDM, making it a suitable candidate for capturing and understanding pasture growth dynamics in the study region. The processing time for the grid search was approximately 72.6 minutes, reflecting the computational effort required to optimise the model.

Hyperparameter	Value
Max Depth	35
No. of Estimators	200
Min Samples Split	10
Min Samples Leaf	1
Random State	42
Bootstrap	True

Table 1- Optimised Hyperparameters for the best Random Forest Regressor model

Train MAPE	Validation MAPE	Test MAPE	Test Accuracy (%) (1 - MAPE)	Processing Time (min)
0.117	0.073	0.177	82.25	72.6

Table 2- Best Random Forest Regressor Model Performance

6.2.2 Feature Importance Interpretation

The feature importances as shown in Figure 9, indicate which variables have the most influence on the pasture growth predictions made by the best Random Forest Regressor model. This analysis provides insights into how different features contribute to predicting TSDM and can guide practical agricultural management strategies to optimise pasture productivity.

- **LANDSIZE_HA**: With the highest feature importance score (0.253), the size of the land (paddock) significantly impacts TSDM. This makes sense because larger paddocks might have different growth dynamics compared to smaller ones. This is a critical consideration for understanding productivity, as land area determines the extent of biomass accumulation.

- **RADIATION:** Solar radiation plays a significant role in driving photosynthesis, which is essential for plant growth. It explains around 18.5% of the model's importance. The energy provided by solar radiation is critical for plant development, and variations in radiation levels could directly influence TSDM predictions.

- **RH_TMIN** and **RAIN**: Minimum relative humidity (0.169) and rainfall (0.145) show substantial influence. These parameters are closely related to water availability and soil moisture content, which are essential for sustaining growth in pasture systems. In semi-arid climates, moisture levels affect plant health and productivity, explaining their high importance in TSDM prediction. This suggests that TSDM is highly sensitive to the lowest humidity levels**.** Lower relative humidity, especially during dry periods, can significantly affect soil moisture and plant transpiration, thus impacting pasture growth. This suggests that the moisture in the air during minimum temperature periods affects pasture growth. Rainfall, with 14.5% contribution, is also an important factor for predicting pasture growth, as water availability is key for photosynthesis and biomass accumulation.

- **Season_Spring**: Spring (0.073) appears as a key seasonal indicator. Typically, spring months bring favourable conditions for growth, such as moderate temperatures and increased rainfall. Identifying Spring as a predictor supports understanding the seasonality aspect, where rapid pasture growth is expected in response to optimal weather conditions.

- **CROP_TYPE** features are relatively low in importance. The highest importance among them is for **natural grasses** (0.007)**,** which implies that pasture species does have some effect but is not a major driver of TSDM compared to other features.

- **Year**: Year shows a relatively lower importance (0.010). This suggests that yearly trends in the dataset, which may be linked to long-term climatic changes, management practices, or other year-on-year factors, slightly influence TSDM.

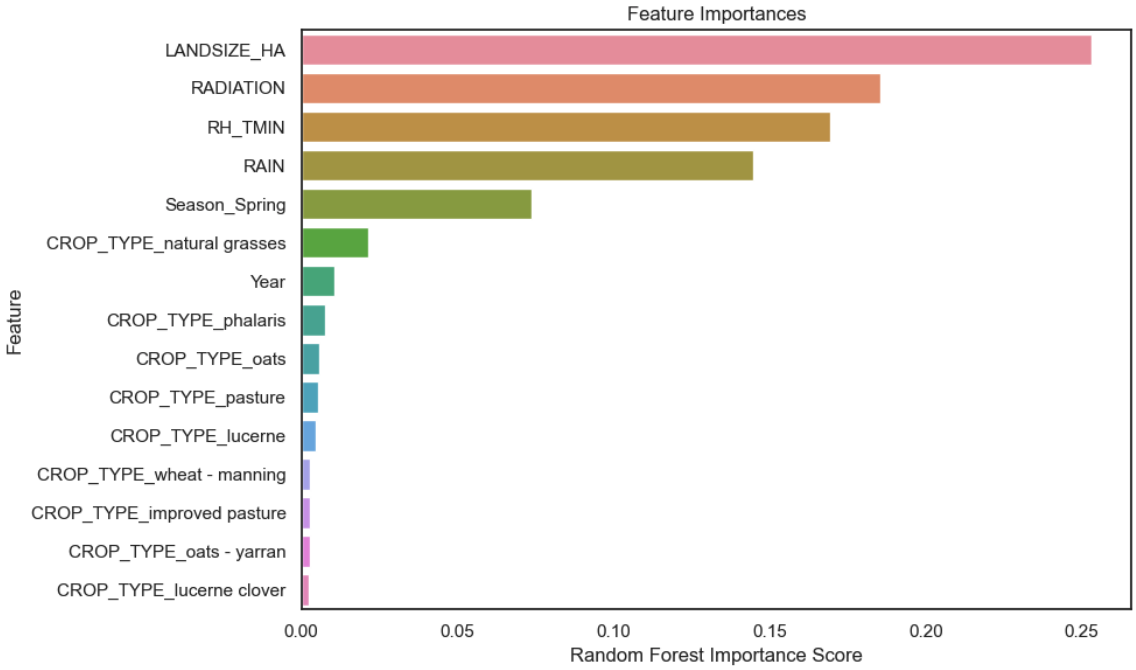

Figure 9- The feature importances plot shows the top 15 variables with the most influence on the pasture growth predictions made by the best Random Forest Regressor model.

6.3. Support Vector Regressor Implementation

To predict Total Standing Dry Matter (TSDM) values, a Support Vector Regression (SVR) model was implemented with a hyperparameter tuning strategy to enhance accuracy and manage data imbalance. The SVR model, with its suitability for handling smaller datasets and capacity to generalise well on agricultural data, was further optimised using GridSearchCV for hyperparameter selection, ensuring the best fit to the TSDM data characteristics.

A grid search was conducted with a range of hyperparameters to optimise SVR. Specifically, the parameter grid included values for C, epsilon, and the kernel type. The C parameter controls the regularisation strength, allowing the model to adjust its tolerance for misclassified points and balance model complexity. The epsilon parameter defines the margin of error, enabling SVR to disregard data points within this margin to enhance tolerance for noise and outliers, a key aspect given the variations within TSDM values. Finally, the kernel parameter, with options for both linear and radial basis function (RBF) kernels, allows the model to select between a simpler, linear relationship or a nonlinear mapping, suited to capture complex patterns in the TSDM data.

A PredefinedSplit approach was applied to evaluate training and validation performance separately. By assigning sample weights inversely proportional to TSDM frequency in train set, the SVR model emphasised underrepresented values, helping to address imbalances in the dataset. The combined training and validation sets, along with sample weighting, allowed the grid search to find optimal parameters that achieved a minimum mean squared error (MSE) score.

The best SVR model, selected from grid search, was then evaluated on validation and test sets using Mean Absolute Percentage Error (MAPE) and Accuracy (calculated as 1 - MAPE). These metrics, expressed as percentages, allow for an intuitive assessment of model performance, providing insight into both prediction error and accuracy. Overall, this SVR implementation, with balanced parameter tuning and weight adjustment for data imbalance, offered a robust solution for TSDM prediction.

6.3.1. Results

The best Support Vector Regressor (SVR) model was obtained through hyperparameter tuning, which identified the optimal settings listed in Table 3. These parameters yielded strong predictive performance across training, validation, and test sets, as summarised in Table 4.

Hyperparameter	Value
C	10
Epsilon	0.5
Kernel	linear

Table 3- Optimised Hyperparameters for the best Support Vector Regressor model

Train MAPE	Validation MAPE	Test MAPE	Test Accuracy (%) (1 - MAPE)	Processing Time (min)
0.497	0.144	0.143	85.73	125.6

Table 4- Best Support Vector Regressor Model Performance

This best SVR model demonstrates robust performance, particularly on the test set, achieving a **test accuracy of 85.73%.** The low MAPE values across validation and test sets indicate reliable prediction accuracy, suggesting that the model generalises well to unseen data. The processing time of 125.6 minutes reflects the computational demands of the SVR with the selected hyperparameters.

6.4. Time-series Forecasting model using LSTM

Since Year and Season influence TSDM, using temporal models that capture seasonality like time-series models with seasonal decomposition (e.g., LSTM) could enhance predictions. To implement LSTM, data was transformed into sequences where each sequence contains observations of multiple seasons. Then, an LSTM network using TensorFlow and Keras frameworks was constructed.

The input shape of (4, 372) allows the model to look back across 4 prior time steps, each containing detailed feature data. This setup enables the model to identify patterns over multiple observations, recognizing recurring patterns influenced by environmental and climate factors that change with the seasons. Each of the 4 time steps can represent different seasonal points, allowing the LSTM to learn these seasonal variations.

The first LSTM layer had 128 units and output sequences to learn detailed time-based dependencies; this was followed by a 20% dropout layer, which prevented overfitting by promoting generalisation. A second LSTM layer with 64 units consolidated these learned patterns into a single vector, retaining the most relevant temporal features. Finally, a dense output layer produced a single TSDM prediction based on the seasonal patterns learned across the sequence, making it suitable for seasonal analysis when paired with well-prepared seasonal data.

The hyperparameters for the Forecasting LSTM model, are presented in the Table 5. The batch size of 32 determines the number of instances processed before updating the model's parameters, balancing computational efficiency and convergence speed. Training the model for 100 epochs ensures sufficient exposure to the data without risking overfitting. The dropout rate of 20% helps prevent overfitting by randomly dropping a portion of neurons

during training, improving generalisation. The Adam optimiser adjusts the learning rate for each parameter and enhances training efficiency and performance. For the loss function, Mean Squared Error is used to guide the model's learning by measuring the average squared difference between predicted and actual values. A validation data was set as defined in data split section, to monitor performance of the model and prevent overfitting during training.

Hyperparameter	Value
Batch size	32
No. of Epochs	100
Optimiser	Adam
Loss	Mean Squared Error
Validation Split	Manually Split

Table 5- Overview of training hyperparameters for the Forecasting LSTM

6.4.1. Results

Figure 10 illustrates the decrease of the MSE loss over epochs for training and validation data set indicating that the model effectively learned the patterns in the data without overfitting.

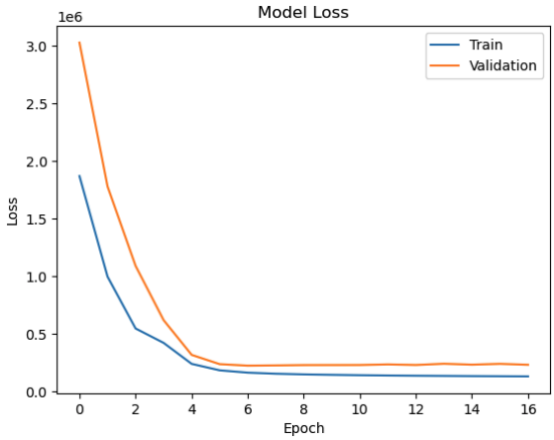

Figure 10- Training loss and validation loss over the epochs for Forecasting LSTM model.

Train MAPE	Validation MAPE	Test MAPE	Test Accuracy (%) (1 - MAPE)	Processing Time (min)
0.234	0.191	0.159	84.11	10.9

Table 6- Forecasting LSTM Model Performance

This model demonstrates robust performance, particularly on the test set, achieving a **test accuracy of 84.11%.** The low MAPE values across validation and test sets indicate reliable prediction accuracy, suggesting that the model generalises well to unseen data, which is essential for real-world seasonal forecasting. The Processing Time of 10.9 minutes suggests that this architecture can be trained efficiently within a reasonable timeframe, making it feasible for iterative tuning and deployment in seasonal pasture forecasting contexts.

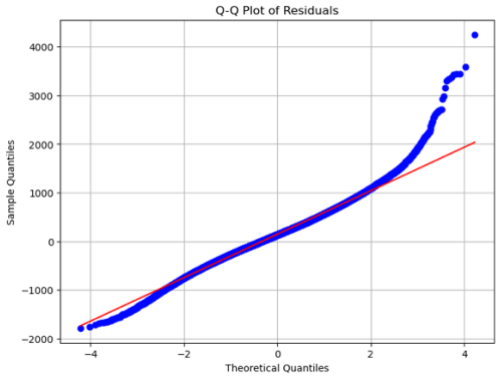

Figure 11- A Q-Q plot for the residuals of the LSTM model predictions.

The Q_Q plot shows the most points are spread around the reference line (which represents a normal distribution) but there are some deviations in the right tail. This suggests that while the bulk of the data may resemble a normal distribution, there are significant high-value outliers or that the data may have a skewed distribution with a long right tail. This could lead to a reassessment of the model's effectiveness and the need for potential adjustments.

6.5. Time-series Forecasting model using LSTM (Complex)

To deepen the deep learning study, another LSTM model was conducted by adding more layer to the previous model. Similarly, the input shape of (4, 372) was used. The first LSTM layer had 256 units and output sequences to learn detailed time-based dependencies, followed by a 20% dropout layer. This was followed by a second LSTM layer with 128 units followed by a 20% dropout layer. The third layer consisting of 64 units consolidated these learned patterns into a single vector, retaining the most relevant temporal features. Finally, a dense output layer produced a single TSDM prediction based on the seasonal patterns learned across the sequence. The hyperparameters settings mirrored those used in the first forecasting LSTM model.

6.5.1. Results

Figure 12 illustrates the decrease of the MSE loss over epochs for training and validation data set indicating that the model effectively learned the patterns in the data without overfitting.

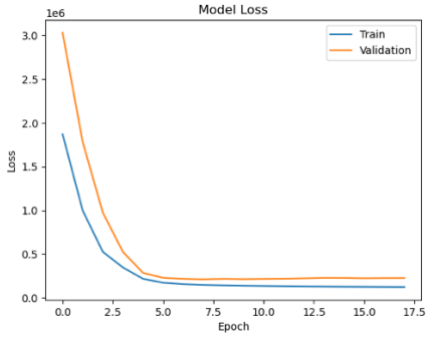

Figure 12- Training loss and validation loss over the epochs for Forecasting LSTM model (Complex).

Train MAPE	Validation MAPE	Test MAPE	Test Accuracy (%) (1 - MAPE)	Processing Time (min)
0.225	0.104	0.161	83.88	24.4

Table 7- Complex Forecasting LSTM Model Performance

The performance metrics for the Complex Forecasting LSTM Model, as shown in Table Y, reveal a promising outcome. The Model showed a good performance, particularly on the validation set, which reflects its ability to generalise well. The lower MAPE in the validation and test phases compared to the training indicates effective learning and a robust model, despite a slightly higher training error. The test MAPE of 0.161 reflects an average prediction error of 16.1% on unseen data, which is better than the training error but slightly higher than the validation error, highlighting the model's robustness despite some variability in performance. Additionally, the processing time of 24.4 minutes provides insight into the computational efficiency of the training process, which may be considered reasonable given the complexity of the model architecture.

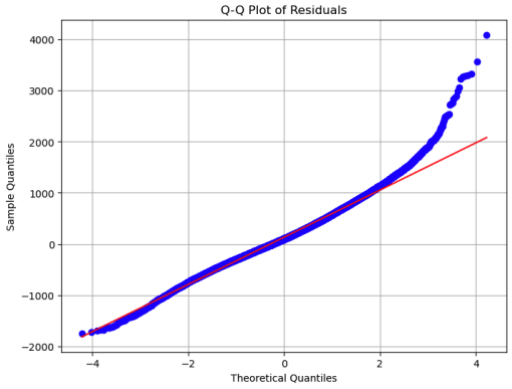

Figure 13- A Q-Q plot for the residuals of the complex forecasting LSTM model predictions.

The Q_Q plot is similar to the simple forecasting LSTM model, with only a slight improvement demonstrated at the left tail.

7. Conclusion and Future Work

Conclusion

The analysis of this study reveals critical insights into the variables influencing Total Standing Dry Matter (TSDM) predictions, providing valuable guidance for agricultural management. The results highlighted that paddock size emerged as the most significant factor, underscoring the importance of land area in determining biomass accumulation and productivity dynamics in pastures. Additionally, solar radiation, a key driver of photosynthesis, accounted for a substantial portion of the model's predictive power, emphasising its vital role in plant growth. relative humidity at minimum temperature and rainfall amount were also identified as influential parameters, reflecting their crucial impact on water availability and soil moisture—factors that are particularly critical in semi-arid environments. The recognition of spring as a key seasonal indicator further reinforces the significance of understanding seasonal patterns in pasture growth, as optimal weather conditions during this period foster rapid biomass accumulation. In contrast, crop type and year demonstrated relatively lower importance, suggesting that while species selection and long-term climatic trends are relevant, they are less critical than immediate environmental conditions and land characteristics. In conclusion, these findings not only address the research question but also offer practical implications for optimising pasture management strategies by focusing on the most influential factors that drive productivity in seasonal contexts.

Model	MAPE	Prediction Accuracy (%) (1 - MAPE)	Processing Time (min)
RF Regressor	0.177	82.25	72.6
SVR	0.143	85.73	125.6
Forecasting LSTM	0.159	84.11	10.9
Forecasting LSTM-Complex	0.161	83.88	24.4

Table 8- The performance comparison between for proposed models.

The performance comparison between the models shows that RF Regressor achieved a Mean Absolute Percentage Error (MAPE) of 0.177, and a prediction accuracy of 82.25%. SVR demonstrated the best performance among the models, with a MAPE of 0.143 and a prediction accuracy of 85.73%. This indicates its strong capability in handling the complexities of TSDM prediction and suggests that SVR is particularly effective in capturing the relationships between the influencing factors and TSDM values (Table 8).

The simple Forecasting LSTM model also performed well, by achieving a MAPE of 0.159 and an accuracy of 84.11%. The more complex Forecasting LSTM model had a slight decrease in performance with a MAPE of 0.161 and an accuracy of 83.88%. Although this indicates that the added complexity did not improve predictive accuracy, it highlights the importance of model selection in the context of agricultural forecasting.
Processing times varied among models, with the RF Regressor taking the longest at 72.6 minutes, followed by SVR at 125.6 minutes, while the Forecasting LSTM models exhibited relatively short training times, with the standard model requiring only 10.9 minutes and the complex model taking 24.4 minutes.

Considering both efficiency and effectiveness, while SVR emerged as the top-performing model, the **Forecasting LSTM model** approach offered efficient training time and acceptable prediction accuracy. In contrast, the SVR model had a longer processing time, which may not be feasible for large datasets or real-time applications. Thus, the Forecasting LSTM model balances reasonable accuracy with computational efficiency, making it a strong candidate for practical applications in agricultural forecasting. This study emphasises the significance of selecting appropriate predictive models tailored to agricultural applications, particularly in forecasting seasonal TSDM.

Reflections on the Conduct of the Project

Throughout this project, several key learning experiences and insights have been learnt. Initially, the project's scope, focused on forecasting Total Standing Dry Matter (TSDM) in agricultural settings, posed unique challenges and opportunities for exploration. The integration of various data sources, including paddock metadata, climate data, and NRM1010 TSDM data, highlighted the importance of data quality and preprocessing in building reliable predictive models. This experience underscored the necessity of thorough data cleaning and transformation, as these steps directly influenced the accuracy of the predictions.

A significant challenge encountered was the issue of imbalanced data, particularly regarding the distribution of TSDM values across different paddocks and seasons. Addressing this imbalance was crucial for ensuring that the models were not biased towards predicting dominant classes. Implementing techniques such as oversampling improved the models' ability to generalise across varied TSDM conditions.

Throughout the modeling phase, the iterative nature of model selection and evaluation became evident. The need to balance model complexity with interpretability was a significant takeaway,

particularly as the results indicated that simpler models could often yield comparable accuracy with less computational demand.

Moreover, collaboration and feedback from mentors regarding feature importance and model performance helped to contextualise the findings within the broader agricultural landscape. Overall, this project has enhanced my understanding of machine learning applications in agriculture and the critical factors influencing model performance in real-world settings.

Future Work

Based on the insights gained, several directions for future work have been identified:

One potential direction is the incorporation of additional data sources, such as remote sensing imagery from satellites and drones such as Landsat and MODIS, which could provide more granular insights into pasture conditions and growth dynamics. By integrating these advanced data types, future models may achieve higher accuracy and robustness in predictions.

Another promising area of investigation lies in the application of hybrid modeling techniques that combine the strengths of multiple algorithms. For instance, ensemble methods that incorporate predictions from various models could potentially improve overall performance and provide more reliable forecasts. Additionally, further research could focus on optimising hyperparameters and conducting extensive cross-validation to ensure the robustness of the models across different datasets and environmental conditions.

Moreover, exploring techniques specifically designed to address imbalanced data in TSDM predictions is crucial. Future research could investigate advanced resampling methods, cost-sensitive learning, or the integration of synthetic data generation to create a more balanced representation of TSDM values across the dataset. This approach would likely improve model training and improve predictive accuracy.

Lastly, Exploring the impact of climate change on pasture growth dynamics is also an essential consideration for future research. Including this into predictive models could help stakeholders make informed decisions regarding pasture management and sustainability in the long term.

8.References

[1] J. N. Brown, A. Ash, N. MacLeod, and P. McIntosh, "Diagnosing the weather and climate features that influence pasture growth in Northern Australia," *Climate Risk Management*, vol. 24, pp. 1–12, 2019, doi: https://doi.org/10.1016/j.crm.2019.01.003.

[2] P. D. Barrett, A. S. Laidlaw, and C. S. Mayne, "GrazeGro: a European herbage growth model to predict pasture production in perennial ryegrass swards for decision support," *European Journal of Agronomy*, vol. 23, no. 1, pp. 37–56, Jul. 2005, doi: https://doi.org/10.1016/j.eja.2004.09.006.
[3] M. Masoud, J. Hsieh, K. Helmstedt, J. McGree, and P. Corry, "An integrated pasture biomass and beef cattle liveweight predictive model under weather forecast uncertainty: An application to Northern Australia," *Food and Energy Security*, Mar. 2023, doi: https://doi.org/10.1002/fes3.453.

[4] D. De Rosa *et al.*, "Predicting pasture biomass using a statistical model and machine learning algorithm implemented with remotely sensed imagery," *Computers and Electronics in Agriculture*, vol. 180, p. 105880, Jan. 2021, doi: https://doi.org/10.1016/j.compag.2020.105880.

[5] M. Jouven, P. Carrere, and R. Baumont, "Model predicting dynamics of biomass, structure and digestibility of herbage in managed permanent pastures. 1. Model description," Grass and Forage Science, vol. 61, no. 2, pp. 112–124, Jun. 2006, doi: https://doi.org/10.1111/j.1365-2494.2006.00515.x.

[6] Guilherme Defalque, R. Santos, Davi Bungenstab, D. Echeverria, A. Dias, and Cristiane Defalque, "Machine learning models for dry matter and biomass estimates on cattle grazing systems," *Computers and Electronics in Agriculture*, vol. 216, pp. 108520–108520, Jan. 2024, doi: https://doi.org/10.1016/j.compag.2023.108520.

[7] M. Rangwala *et al.*, "DeepPaSTL: Spatio-Temporal Deep Learning Methods for Predicting Long-Term Pasture Terrains Using Synthetic Datasets," *Agronomy*, vol. 11, no. 11, p. 2245, Nov. 2021, doi: https://doi.org/10.3390/agronomy11112245.

[8] C. Stumpe, Joerg Leukel, and T. Zimpel, "Prediction of pasture yield using machine learning-based optical sensing: a systematic review," *Precision Agriculture*, vol. 25, no. 1, pp. 430–459, Sep. 2023, doi: https://doi.org/10.1007/s11119-023-10079-9.

[9] M. Correa-Luna *et al.*, "Accounting for minimum data required to train a machine learning model to accurately monitor Australian dairy pastures using remote sensing," *Scientific Reports*, vol. 14, no. 1, p. 16927, Jul. 2024, doi: https://doi.org/10.1038/s41598-024-68094-3.

[10] C. Nickmilder *et al.*, "Development of Machine Learning Models to Predict Compressed Sward Height in Walloon Pastures Based on Sentinel-1, Sentinel-2 and Meteorological Data Using Multiple Data Transformations," *Remote Sensing*, vol. 13, no. 3, p. 408, Jan. 2021, doi: https://doi.org/10.3390/rs13030408.

[11] E. M. Morse-McNabb, Md Farhad Hasan, and S. Karunaratne, "A Multi-Variable Sentinel-2 Random Forest Machine Learning Model Approach to Predicting Perennial Ryegrass Biomass in Commercial Dairy Farms in Southeast Australia," *Remote sensing*, vol. 15, no. 11, pp. 2915–2915, Jun. 2023, doi: https://doi.org/10.3390/rs15112915.

[12] C. Nickmilder *et al.*, "Creation of a Walloon Pasture Monitoring Platform Based on Machine Learning Models and Remote Sensing," *Remote Sensing*, vol. 15, no. 7, pp. 1890–1890, Mar. 2023, doi: https://doi.org/10.3390/rs15071890.

[13] Y. Chen, J. Guerschman, Y. Shendryk, D. Henry, and M. T. Harrison, "Estimating Pasture Biomass Using Sentinel-2 Imagery and Machine Learning," *Remote Sensing*, vol. 13, no. 4, p. 603, Feb. 2021, doi: https://doi.org/10.3390/rs13040603.

[14] A. Guevara-Escobar, M. Cervantes-Jiménez, V. Lemus-Ramírez, A. K. Yabuta-Osorio, and J. G. García-Muñiz, "Estimation of forage mass in a mixed pasture by machine learning, pasture management and satellite meteorological data," *Revista Mexicana de Ciencias Pecuarias*, vol. 14, no. 1, pp. 61–77, Dec. 2022, doi: https://doi.org/10.22319/rmcp.v14i1.6162.

[15] "NRM Regions Australia – NRM Regions Australia is the national collaborative group for Australia's 56 regional NRM organisations." https://nrmregionsaustralia.com.au/

[16] M. H. M. da R. Fernandes, J. de S. FernandesJunior, J. M. Adams, M. Lee, R. A. Reis, and L. O. Tedeschi, "Using sentinel-2 satellite images and machine learning algorithms to predict tropical pasture forage mass, crude protein, and fiber content," *Scientific Reports*, vol. 14, no. 1, p. 8704, Apr. 2024, doi: https://doi.org/10.1038/s41598-024-59160-x.

[17] Z. Zhou, J. Morel, D. Parsons, Sergey Kucheryavskiy, and A. Gustavsson, "Estimation of yield and quality of legume and grass mixtures using partial least squares and support vector machine analysis of spectral data," *Computers and Electronics in Agriculture*, vol. 162, pp. 246–253, Jul. 2019, doi: **https://doi.org/10.1016/j.compag.2019.03.038**.

[18] S. N. Subhashree *et al.*, "Tools for Predicting Forage Growth in Rangelands and Economic Analyses—A Systematic Review," *Agriculture*, vol. 13, no. 2, p. 455, Feb. 2023, doi: https://doi.org/10.3390/agriculture13020455.

[19] C. Ou *et al.*, "Using Machine Learning Methods Combined with Vegetation Indices and Growth Indicators to Predict Seed Yield of Bromus inermis," *Plants*, vol. 13, no. 6, pp. 773–773, Mar. 2024, doi: **https://doi.org/10.3390/plants13060773**.

[20] M. H. M. da R. Fernandes, J. de S. FernandesJunior, J. M. Adams, M. Lee, R. A. Reis, and L. O. Tedeschi, "Using sentinel-2 satellite images and machine learning algorithms to predict tropical pasture forage mass, crude protein, and fiber content," *Scientific Reports*, vol. 14, no. 1, p. 8704, Apr. 2024, doi: https://doi.org/10.1038/s41598-024-59160-x.

Appendices

Appendix A

Data Loading, Exploration, preprocessing , and Integration:

1.1 nrm1010_tsdm

```
import pandas as pd

# Load the datasets
nrm1010_tsdm_df = pd.read_csv('Data/nrm1010_tsdm.csv')
nrm1010_tsdm_df
```

```
#Convert PADDOCK_ID column to Lowercase String
nrm1010_tsdm_df['PADDOCK_ID'] = nrm1010_tsdm_df['PADDOCK_ID'].astype('str').str.lower()

#Convert OBSERVATION_DATE column to Date type
import numpy as np
nrm1010_tsdm_df['OBSERVATION_DATE'] = pd.to_datetime(
    nrm1010_tsdm_df['OBSERVATION_DATE'], format='%d/%m/%Y', errors='coerce'
)

nrm1010_tsdm_df['OBSERVATION_DATE'] = nrm1010_tsdm_df['OBSERVATION_DATE'].astype('datetime64[ns]')
```

```
nrm1010_tsdm_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030320 entries, 0 to 1030319
Data columns (total 3 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   PADDOCK_ID        1030320 non-null  object
 1   OBSERVATION_DATE  1030320 non-null  datetime64[ns]
 2   TSDM_MEAN         1011265 non-null  float64
dtypes: datetime64[ns](1), float64(1), object(1)
```

```
# Create 'Year' column by extracting the year from 'OBSERVATION_DATE'
nrm1010_tsdm_df['Year'] = nrm1010_tsdm_df['OBSERVATION_DATE'].dt.year

# Create a function to determine the season
def get_season(month):
    if month in [12, 1, 2]:
        return 'Summer'
    elif month in [3, 4, 5]:
        return 'Autumn'
    elif month in [6, 7, 8]:
        return 'Winter'
    elif month in [9, 10, 11]:
        return 'Spring'

# Apply the function to create a 'Season' column
nrm1010_tsdm_df['Season'] = nrm1010_tsdm_df['OBSERVATION_DATE'].dt.month.apply(get_season)

paddock_ids_with_null_tsdm_mean = nrm1010_tsdm_df[nrm1010_tsdm_df['TSDM_MEAN'].isnull()]['PADDOCK_ID']
paddock_ids_with_null_tsdm_mean.value_counts() #there are 2800 unique paddocks with null tsdm

# Group the data by PADDOCK_ID and check if all TSDM_MEAN values are null for each paddock
paddocks_all_null = nrm1010_tsdm_df.groupby('PADDOCK_ID')['TSDM_MEAN'].apply(lambda x: x.isnull().all())

# Filter to get only those paddocks where all TSDM_MEAN values are null
paddocks_with_all_null_tsdm = paddocks_all_null[paddocks_all_null].index

# find all rows in the nrm1010_tsdm_df dataframe where the PADDOCK_ID is in the paddocks_with_all_null_tsdm list:

# Filter rows where PADDOCK_ID is in paddocks_with_all_null_tsdm
nrm1010_with_all_null_tsdm = nrm1010_tsdm_df[nrm1010_tsdm_df['PADDOCK_ID'].isin(paddocks_with_all_null_tsdm)]

# Filter rows where PADDOCK_ID is in paddocks_with_all_null_tsdm
paddock_metadata_df_with_all_null_tsdm = paddock_metadata_df[paddock_metadata_df['PADDOCK_ID'].isin(paddocks_with_all_null_tsdm)]
# Filter rows where PADDOCK_ID is in paddocks_with_all_null_tsdm
climate_df_with_all_null_tsdm = climate_df[climate_df['PADDOCK_ID'].isin(paddocks_with_all_null_tsdm)]


# Drop rows where PADDOCK_ID is in paddocks_with_all_null_tsdm
nrm1010_tsdm_df = nrm1010_tsdm_df[~nrm1010_tsdm_df['PADDOCK_ID'].isin(paddocks_with_all_null_tsdm)]

# replace the remaining Nan value TSDMs with zero
nrm1010_tsdm_df['TSDM_MEAN'] = nrm1010_tsdm_df['TSDM_MEAN'].fillna(0)

# Step 1: Group by 'PADDOCK_ID', and 'Season' and calculate the mean TSDM_MEAN
mean_TSDM_MEAN_by_paddock = nrm1010_tsdm_df.groupby(['PADDOCK_ID', 'Season'])['TSDM_MEAN'].mean()
```

```
# Step 2: Define a function to replace zero TSDM_MEAN values with the mean for the corresponding paddock
def replace_zero_TSDM_with_mean(row):
    if row['TSDM_MEAN'] == 0:
        # Use a tuple to access the mean value in the MultiIndex Series (excluding 'TSDM_MEAN' from the tuple)
        return mean_TSDM_MEAN_by_paddock.get((row['PADDOCK_ID'], row['Season']), row['TSDM_MEAN']) #'Year', 'Month',
    else:
        return row['TSDM_MEAN']


nrm1010_tsdm_df["TSDM_MEAN_Interpolate"] = nrm1010_tsdm_df["TSDM_MEAN"]

# Step 3: Apply the function to replace zero TSDM_MEAN values
nrm1010_tsdm_df["TSDM_MEAN_Interpolate"] = nrm1010_tsdm_df.apply(replace_zero_TSDM_with_mean, axis=1)


# Group by 'PADDOCK_ID', 'Year' and 'Season' and calculate the mean of 'TSDM_MEAN'
nrm_year_season_grouped = nrm1010_tsdm_df.groupby(['PADDOCK_ID', 'Year', 'Season'])['TSDM_MEAN_Interpolate'].mean() #'Month',

# Convert the result to a DataFrame for a clearer view
nrm_year_season_grouped_df = nrm_year_season_grouped.reset_index()

# Display the result
print(nrm_year_season_grouped_df)
```

Paddock metadata:

```
paddock_metadata_df = pd.read_csv('Data/paddock_metadata.csv')

paddock_metadata_df['CROP_TYPE'] = paddock_metadata_df['CROP_TYPE'].replace(np.nan,'Unknown')

#Convert PADDOCK_ID columns to lower case
paddock_metadata_df['PADDOCK_ID'] = paddock_metadata_df['PADDOCK_ID'].astype('str').str.lower()
nrm1010_tsdm_df['PADDOCK_ID'] = nrm1010_tsdm_df['PADDOCK_ID'].astype('str').str.lower()

paddock_metadata_Grazing_df = paddock_metadata_df[paddock_metadata_df['PASTURE_STATE']== 'Grazing']
```

Climate data:

1.3 Climate data

```
import pandas as pd
import glob
import os

# Path to the folder containing the CSV files
folder_path = 'Data'

# Use glob to find all CSV files in the specified folder
csv_files = glob.glob(os.path.join(folder_path, 'climate_*.csv'))

# Initialize an empty list to store DataFrames
df_list = []

# Loop through the CSV files and read them into DataFrames
for file in csv_files:
    df = pd.read_csv(file)
    df_list.append(df)

# Concatenate all DataFrames into a single DataFrame
climate_df = pd.concat(df_list, ignore_index=True)

# Display the combined DataFrame
print(climate_df)

# Optionally, check the shape to confirm the size of the combined DataFrame
print(climate_df.shape)
```

```
#Convert PADDOCK_ID column to Lowercase String
climate_df['PADDOCK_ID'] = climate_df['PADDOCK_ID'].astype('str').str.lower()
climate_df['DATE'] = climate_df['DATE'].astype('datetime64[ns]')
# Create 'Year' column by extracting the year from 'DATE'
climate_df['Year'] = climate_df['DATE'].dt.year
# Apply the function to create a 'Season' column
climate_df['Season'] = climate_df['DATE'].dt.month.apply(get_season)

# Check for missing values
print(climate_df.isnull().sum())
#---
# Check if there are any zero values in the entire DataFrame except 'RAIN'
any_zeros_except_rain = (climate_df.drop(columns=['RAIN']) == 0).any().any()

# Identify columns with zero values, excluding 'RAIN'
columns_with_zeros_except_rain = climate_df.drop(columns=['RAIN']).columns[
    (climate_df.drop(columns=['RAIN']) == 0).any()
]
#---

## Check Evap = 0 :
climate_df[(climate_df['MIN_TEMP'] > 0) & (climate_df['EVAP'] == 0)].shape[0] #75,681 rows
climate_df[climate_df['EVAP'] == 0].shape[0]  #103,475 rows
climate_df[(climate_df['MIN_TEMP'] <= 0) & (climate_df['EVAP'] == 0)].shape[0] #27,794 rows
#---
climate_df[(climate_df['RAIN'] > 0) & (climate_df['EVAP'] == 0)].shape #88,290 rows
climate_df[(climate_df['RAIN'] == 0) & (climate_df['EVAP'] == 0)].shape #15185 rows

#EVAP Interpolation:
# Step 1: Group by 'PADDOCK_ID', 'Year', 'Season' and calculate the mean EVAP
mean_evap_by_paddock = climate_df.groupby(['PADDOCK_ID', 'Year', 'Season'])['EVAP'].mean()

#To calculate more specific EVAP_Interpolate values, we consider 'PADDOCK_ID', 'Year', 'Season':
# Step 2: Define a function to replace zero EVAP values with the mean for the corresponding paddock
def replace_zero_with_mean(row):
    if row['EVAP'] == 0:
        # Use a tuple to access the mean value in the MultiIndex Series
        return mean_evap_by_paddock.get((row['PADDOCK_ID'], row['Year'], row['Season']), row['EVAP']) #row['Month'],
        #return mean_evap_by_paddock[row['PADDOCK_ID', 'Year', 'Month', 'Season']]
    else:
        return row['EVAP']

climate_df['EVAP_Interpolate'] = climate_df['EVAP']
#climate_df.filter(climate_df['EVAP_Interpolate'] == 0)

# Step 3: Apply the function to replace zero EVAP values
climate_df['EVAP_Interpolate'] = climate_df.apply(replace_zero_with_mean, axis=1)
#---
# Group by 'PADDOCK_ID', 'Year', and 'Season'
grouped_climate_df = climate_df.groupby(['PADDOCK_ID', 'Year', 'Season']).mean().reset_index()
```

Data Integration:

```
merged_df_1 = pd.merge(nrm_year_season_grouped_df, paddock_metadata_Grazing_df, on='PADDOCK_ID', how='inner')
merged_df = pd.merge(merged_df_1, grouped_climate_df, on=['PADDOCK_ID', 'Year', 'Season'], how='inner')

# Drop the 'PASTURE_STATE' and 'CREATION_DATE' columns from merged_df
merged_df = merged_df.drop(columns=['PASTURE_STATE', 'CREATION_DATE'])

# Remove repeated rows
df_cleaned = merged_df.drop_duplicates()
# reset the index
df_cleaned.reset_index(drop=True, inplace=True)
merged_df = df_cleaned
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115176 entries, 0 to 115175
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   PADDOCK_ID             115176 non-null  object
 1   Year                   115176 non-null  int64
 2   Season                 115176 non-null  object
 3   TSDM_MEAN_Interpolate  115176 non-null  float64
 4   CROP_TYPE              115176 non-null  object
 5   LANDSIZE_HA            115176 non-null  float64
 6   RAIN                   115176 non-null  float64
 7   MAX_TEMP               115176 non-null  float64
 8   MIN_TEMP               115176 non-null  float64
 9   RH_TMAX                115176 non-null  float64
 10  RH_TMIN                115176 non-null  float64
 11  EVAP                   115176 non-null  float64
 12  RADIATION              115176 non-null  float64
 13  EVAP_Interpolate       115176 non-null  float64
dtypes: float64(10), int64(1), object(3)
```

Appendix B

Feature selection:

Correlation Matrix:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Define feature columns
feature_columns = ['Year', 'Season', 'CROP_TYPE', 'LANDSIZE_HA', 'RAIN', 'MAX_TEMP',
                   'MIN_TEMP', 'RH_TMAX', 'RH_TMIN', 'RADIATION', 'EVAP_Interpolate']


# Separate features and target
X = merged_df[feature_columns]
y = merged_df['TSDM_MEAN_Interpolate']

# Define which columns are categorical and which are numeric
categorical_columns = ['Season', 'CROP_TYPE']
numeric_columns = ['Year', 'LANDSIZE_HA', 'RAIN', 'MAX_TEMP', 'MIN_TEMP',
                   'RH_TMAX', 'RH_TMIN', 'RADIATION', 'EVAP_Interpolate']

#Create a subset of the dataframe with only numeric columns:
combined_df = pd.concat([X, y], axis=1)
df_num = combined_df.select_dtypes(include=[np.number])

import matplotlib.pyplot as plt
#Creating a correlation matrix for continuous variables:
corr = df_num.corr()
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(df_num.columns),1)
ax.set_xticks(ticks)
plt.xticks(rotation=90)
ax.set_yticks(ticks)
ax.set_xticklabels(df_num.columns)
ax.set_yticklabels(df_num.columns)
plt.show()
```

	Feature	Importance
1	LANDSIZE_HA	0.253152
4	RADIATION	0.185444
3	RH_TMIN	0.169226
2	RAIN	0.144537
5	Season_Spring	0.073684
202	CROP_TYPE_natural grasses	0.021243
0	Year	0.010422
280	CROP_TYPE_phalaris	0.007405
230	CROP_TYPE_oats	0.005520
264	CROP_TYPE_pasture	0.005330
145	CROP_TYPE_lucerne	0.004443
359	CROP_TYPE_wheat - manning	0.002874
128	CROP_TYPE_improved pasture	0.002763
239	CROP_TYPE_oats - yarran	0.002536
154	CROP_TYPE_lucerne clover	0.002319

Dimensionality Reduction:

```
# Define feature columns
feature_columns = ['Year', 'Season', 'CROP_TYPE', 'LANDSIZE_HA', 'RAIN', 'RH_TMIN', 'RADIATION']

# Separate features and target
X = merged_df[feature_columns]
y = merged_df['TSDM_MEAN_Interpolate']

# Define which columns are categorical and which are numeric
categorical_columns = ['Season', 'CROP_TYPE']
numeric_columns = ['Year', 'LANDSIZE_HA', 'RAIN', 'RH_TMIN', 'RADIATION']
```

One-hot encoding, standardisation, and Split data:

```
# Use one-hot encoding to convert categorical variables to numeric
features_encoded = pd.get_dummies(X, columns=categorical_columns, drop_first=True)
# Check if there are still non-numeric columns
non_numeric_columns_after_encoding = features_encoded.select_dtypes(include=['object', 'datetime64']).columns
print("Non-numeric columns after encoding:", non_numeric_columns_after_encoding)

feature_names = features_encoded.columns  # Keep this list before converting

from sklearn.preprocessing import StandardScaler
# Normalization or Standardization
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features_encoded)
#create the DataFrame using the correct number of columns
X_transformed_df = pd.DataFrame(features_scaled, columns=feature_names)

# Now, split the data based on 'Year'
X_train = X_transformed_df[merged_df['Year'].between(2017, 2020)]
X_val = X_transformed_df[merged_df['Year'] == 2021]
X_test = X_transformed_df[merged_df['Year'] == 2022]

# Display the sizes of each split
print(f"Training set size: {X_train.shape}")
print(f"Validation set size: {X_val.shape}")
print(f"Test set size: {X_test.shape}")

# Split the target variable in the same way
y_train = y[merged_df['Year'].between(2017, 2020)]
y_val = y[merged_df['Year'] == 2021]
y_test = y[merged_df['Year'] == 2022]
#---
# Set the style and context for a more aesthetic plot
sns.set(style="white")
plt.figure(figsize=(8, 6))

# Plot the histograms with seaborn styling
plt.hist(y_train, bins=50, alpha=0.6, label='Train', color='steelblue', edgecolor='black')
plt.hist(y_val, bins=50, alpha=0.6, label='Validation', color='darkorange', edgecolor='black')
plt.hist(y_test, bins=50, alpha=0.6, label='Test', color='green', edgecolor='black')

# Add title and labels with better formatting
plt.title('Distribution of TSDM Mean Across Train, Validation, and Test Sets', fontsize=12, pad=10)
plt.xlabel('TSDM Mean (kg DM/ha)', fontsize=10)
plt.ylabel('Frequency', fontsize=10)

# Customize legend
plt.legend(loc='upper right', fontsize=10)

# Adjust layout and show plot
plt.tight_layout()
plt.show()
```

Appendix C

Proposed Models:

Random Forest Regressor:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import PredefinedSplit, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

# Define sample weights inversely proportional to frequency in y_train
# This helps balance low-frequency values in TSDM_MEAN_Interpolate
weight = 1.0 / (np.histogram(y_train, bins='auto')[0] + 1)
sample_weight = np.interp(y_train, np.histogram(y_train, bins='auto')[1][:-1], weight)

# Concatenate X_train and X_val for combined training
X_combined = pd.concat([X_train, X_val])
y_combined = np.concatenate([y_train, y_val])

# Define the PredefinedSplit indices
train_val_split_index = np.array([0] * len(y_train) + [-1] * len(y_val))
ps = PredefinedSplit(train_val_split_index)

# Define the parameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [25, 30, 35, None],
    'min_samples_split': [8, 10, 12],
    'min_samples_leaf': [1, 2, 3],
    'bootstrap': [True, False]
}

# Initialize RandomForestRegressor model
rf_model = RandomForestRegressor(random_state=42)

# Initialize GridSearchCV with PredefinedSplit and sample weighting
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=ps,
                           scoring='neg_mean_squared_error', verbose=0)

# Fit the grid search with combined data and sample weights
grid_search.fit(X_combined, y_combined, sample_weight=np.concatenate([sample_weight, np.ones(len(y_val))]))
#*****************************************************************************************************
# Get the best parameters and the best model
best_params = grid_search.best_params_
best_rf = grid_search.best_estimator_

# Print the best parameters and best score
print(f"Best Parameters: {best_params}")
print("Best score:", grid_search.best_score_)

# Evaluate the best model on the validation set
y_val_pred = best_rf.predict(X_val)
```

```python
# Calculate performance metrics on the validation set
mse_val = mean_squared_error(y_val, y_val_pred)
r2_val = r2_score(y_val, y_val_pred)
print(f"Validation Set: MSE (Best Model): {mse_val}")
print(f"Validation Set: R² (Best Model): {r2_val}")
mape_val = mean_absolute_percentage_error(y_val, y_val_pred)
accuracy_val = (1 - mape_val) * 100  # Convert to percentage
print(f"Validation Set: Mean Absolute Percentage Error (MAPE) (Best Model): {mape_val}")
print(f"Validation Set: Prediction Accuracy (Best Model): {accuracy_val:.2f}%")


# Once validated, evaluate the best model on the test set
y_test_pred = best_rf.predict(X_test)

# Calculate performance metrics on the test set
mse_test = mean_squared_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)
print(f"Test Set: MSE (Best Model): {mse_test}")
print(f"Test Set: R² (Best Model): {r2_test}")
# MAPE
mape = mean_absolute_percentage_error(y_test, y_test_pred)
# Accuracy as (1 - MAPE)
accuracy = (1 - mape) * 100  # Convert to percentage
print(f"Test Set: Mean Absolute Percentage Error (MAPE): {mape}")
print(f"Test Set: Prediction Accuracy: {accuracy:.2f}%")
```

```
Best Parameters: {'bootstrap': True, 'max_depth': 35, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200}
Best score: -511095.25049860496
Validation Set: MSE (Best Model): 37667.59638504616
Validation Set: R² (Best Model): 0.7770727972035046
Validation Set: Mean Absolute Percentage Error (MAPE) (Best Model): 0.07309802565119203
Validation Set: Prediction Accuracy (Best Model): 92.69%
Test Set: MSE (Best Model): 284451.2609116557
Test Set: R² (Best Model): -0.9286971452150419
Test Set: Mean Absolute Percentage Error (MAPE): 0.17747414815395826
Test Set: Prediction Accuracy: 82.25%
```

```python
y_pred_best_train = best_rf.predict(X_train)
# MAPE
mape_train = mean_absolute_percentage_error(y_train, y_pred_best_train)
# Accuracy as (1 - MAPE)
accuracy_train = (1 - mape_train) * 100  # Convert to percentage
print(f"Train Set: Mean Absolute Percentage Error (MAPE): {mape_train}")
print(f"Train Set: Prediction Accuracy: {accuracy_train:.2f}%")
```

```
Train Set: Mean Absolute Percentage Error (MAPE): 0.11749573883998454
Train Set: Prediction Accuracy: 88.25%
```

Support Vector Regressor:

```
from sklearn.model_selection import PredefinedSplit, GridSearchCV
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error
import numpy as np
import pandas as pd

# Define the parameter grid for SVR
param_grid = {
    'C': [0.1, 1, 10],
    'epsilon': [0.1, 0.2, 0.5],
    'kernel': ['linear', 'rbf']
}

# Calculate sample weights inversely proportional to frequency in y_train
weight = 1.0 / (np.histogram(y_train, bins='auto')[0] + 1)
sample_weight = np.interp(y_train, np.histogram(y_train, bins='auto')[1][:-1], weight)

# Combine training and validation target variables into one array
y_combined = np.concatenate([y_train, y_val])

# Create the indices for PredefinedSplit
# -1 indicates that those samples are used for validation (i.e., X_val)
# 0 indicates that those samples are used for training (i.e., X_train)
train_val_split_index = np.array([0] * len(y_train) + [-1] * len(y_val))

# Initialize PredefinedSplit object
ps = PredefinedSplit(train_val_split_index)

# Initialize the SVR model
svr = SVR()

# Initialize GridSearchCV with the predefined split
grid_search = GridSearchCV(estimator=svr, param_grid=param_grid, cv=ps, scoring='neg_mean_squared_error', verbose=0)

# Combine training and validation sets for fitting
X_combined = pd.concat([X_train, X_val])

# Fit the GridSearchCV model using sample weights for the training set and default weights (1.0) for validation set
grid_search.fit(X_combined, y_combined, sample_weight=np.concatenate([sample_weight, np.ones(len(y_val))]))
#*****************************************************************************************************

# Get the best parameters and the best model
best_params = grid_search.best_params_
best_svr = grid_search.best_estimator_

# Print the best parameters and score
print(f"Best Parameters: {best_params}")
print("Best Score (MSE):", -grid_search.best_score_)
```

```
# Evaluate the best model on the validation set
y_val_pred = best_svr.predict(X_val)

# Calculate performance metrics on the validation set
mse_val = mean_squared_error(y_val, y_val_pred)
r2_val = r2_score(y_val, y_val_pred)
print(f"Validation Set: MSE (Best Model): {mse_val}")
print(f"Validation Set: R² (Best Model): {r2_val}")
mape_val = mean_absolute_percentage_error(y_val, y_val_pred)
# Accuracy as (1 - MAPE)
accuracy_val = (1 - mape_val) * 100  # Convert to percentage
print(f"Validation Set: Mean Absolute Percentage Error (MAPE) (Best Model): {mape_val}")
print(f"Validation Set: Prediction Accuracy (Best Model): {accuracy_val:.2f}%")

# Once validated, evaluate the best model on the test set
y_test_pred = best_svr.predict(X_test)

# Calculate performance metrics on the test set
mse_test = mean_squared_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)
print(f"Test Set: MSE (Best Model): {mse_test}")
print(f"Test Set: R² (Best Model): {r2_test}")
# MAPE
mape = mean_absolute_percentage_error(y_test, y_test_pred)
# Accuracy as (1 - MAPE)
accuracy = (1 - mape) * 100  # Convert to percentage
print(f"Test Set: Mean Absolute Percentage Error (MAPE): {mape}")
print(f"Test Set: Prediction Accuracy: {accuracy:.2f}%")
#*************************************************************************************************************
```

```
Best Parameters: {'C': 10, 'epsilon': 0.5, 'kernel': 'linear'}
Best Score (MSE): 463685.7771873516
Validation Set: MSE (Best Model): 141727.23997857078
Validation Set: R² (Best Model): 0.16121918570218619
Validation Set: Mean Absolute Percentage Error (MAPE) (Best Model): 0.1439740314551151
Validation Set: Prediction Accuracy (Best Model): 85.60%
Test Set: MSE (Best Model): 186386.9977699758
Test Set: R² (Best Model): -0.263780899940544
Test Set: Mean Absolute Percentage Error (MAPE): 0.14267455946836707
Test Set: Prediction Accuracy: 85.73%
```

```
y_pred_best_train = best_svr.predict(X_train)
# MAPE
mape_train = mean_absolute_percentage_error(y_train, y_pred_best_train)
# Accuracy as (1 - MAPE)
accuracy_train = (1 - mape_train) * 100  # Convert to percentage
print(f"Train Set: Mean Absolute Percentage Error (MAPE): {mape_train}")
print(f"Train Set: Prediction Accuracy: {accuracy_train:.2f}%")
```

```
Train Set: Mean Absolute Percentage Error (MAPE): 0.49696607065113313
Train Set: Prediction Accuracy: 50.30%
```

Forecasting LSTM:

Steps to Prepare Data for LSTM

```
target = 'TSDM_MEAN_Interpolate'
y = merged_df[target].values  # Extract the target variable as a NumPy array

# Now create the DataFrame using the correct number of columns
X_transformed_df = pd.DataFrame(features_scaled, columns=feature_names)

def create_sequences(data, target, sequence_length=4):  # 4 timesteps
    sequences = []
    targets = []

    for i in range(len(data) - sequence_length):
        seq = data.iloc[i:i + sequence_length].values  # Use all features for the sequence
        label = target[i + sequence_length]  # Access target directly from the array
        sequences.append(seq)
        targets.append(label)

    return np.array(sequences), np.array(targets)

# Create sequences
sequence_length = 4  # use 4 seasonal timesteps
X, y = create_sequences(X_transformed_df, y, sequence_length)

# Reshape for LSTM: [samples, timesteps, features]
print(f"Shape of X (input): {X.shape}")
print(f"Shape of y (target): {y.shape}")
```

```
Shape of X (input): (345524, 4, 372)
Shape of y (target): (345524,)
```

```
# Now, split the data based on 'Year'
train_mask = X_transformed_df[merged_df['Year'].between(2017, 2020)]
val_mask = X_transformed_df[merged_df['Year'] == 2021]
test_mask = X_transformed_df[merged_df['Year'] == 2022]

target = 'TSDM_MEAN_Interpolate'
y = merged_df[target].values  # Extract the target variable as a NumPy array

# Split the target variable in the same way
y_train_mask = y[merged_df['Year'].between(2017, 2020)]
y_val_mask = y[merged_df['Year'] == 2021]
y_test_mask = y[merged_df['Year'] == 2022]

X_train, y_train = create_sequences(train_mask, y_train_mask, sequence_length)
X_val, y_val = create_sequences(val_mask, y_val_mask, sequence_length)
X_test, y_test = create_sequences(test_mask, y_test_mask, sequence_length)

# Display the sizes of each split
print(f"Training set size: {X_train.shape}")
print(f"Validation set size: {X_val.shape}")
print(f"Test set size: {X_test.shape}")
```

```
Training set size: (230348, 4, 372)
Validation set size: (57584, 4, 372)
Test set size: (57584, 4, 372)
```

Forecasting LSTM Model:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import tensorflow.keras as keras
from tensorflow.keras.callbacks import EarlyStopping
import random

# Set the random seed for reproducibility
random_seed = 42
np.random.seed(random_seed)
random.seed(random_seed)
tf.random.set_seed(random_seed)

# Define EarlyStopping callback to stop training if validation loss stops improving
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Build LSTM model
model = Sequential()

# Adding the LSTM layer with 128 units
model.add(LSTM(units=128, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))

# Adding a Dropout layer to prevent overfitting
model.add(Dropout(0.2))

# Adding another LSTM layer (optional)
model.add(LSTM(units=64, return_sequences=False))

# Adding a final Dense layer for output
model.add(Dense(units=1))  # # Output layer for regression: predicting a single value (TSDM_MEAN)

# Compile the model
#opt = keras.optimizers.Adam(learning_rate=0.001) #1e-3
opt = keras.optimizers.Adam()
model.compile(optimizer=opt, loss='mse')  # Use mean squared error for the loss function

# Summary of the model
print(model.summary())

# Train the model
history = model.fit(X_train, y_train,
          epochs=100,
          batch_size=32,
          validation_data=(X_val, y_val),
          callbacks=[early_stopping],
          verbose=2)
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4, 128)            256512

 dropout (Dropout)           (None, 4, 128)            0

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 305,985
Trainable params: 305,985
Non-trainable params: 0
```

```python
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_percentage_error

# Evaluate the model on the validation set
y_val_pred = model.predict(X_val)
# Calculate performance metrics on the validation set
mse_val = mean_squared_error(y_val, y_val_pred)
r2_val = r2_score(y_val, y_val_pred)
print(f"Validation Set: MSE (Best Model): {mse_val}")
print(f"Validation Set: R² (Best Model): {r2_val}")
# MAPE
mape_val = mean_absolute_percentage_error(y_val, y_val_pred)
# Accuracy as (1 - MAPE)
accuracy_val = (1 - mape_val) * 100  # Convert to percentage
print(f"Validation Set: Mean Absolute Percentage Error (MAPE) (Best Model): {mape_val}")
print(f"Validation Set: Prediction Accuracy (Best Model): {accuracy_val:.2f}%")

# Once validated, evaluate the best model on the test set Make predictions
y_pred = model.predict(X_test)
# Calculate performance metrics on the test set
mse_test = mean_squared_error(y_test, y_pred)
r2_test = r2_score(y_test, y_pred)
print(f"Test Set: MSE (Best Model): {mse_test}")
print(f"Test Set: R² (Best Model): {r2_test}")
# MAPE
mape = mean_absolute_percentage_error(y_test, y_pred)
# Accuracy as (1 - MAPE)
accuracy = (1 - mape) * 100  # Convert to percentage

print(f"Test Set: Mean Absolute Percentage Error (MAPE): {mape}")
print(f"Test Set: Prediction Accuracy: {accuracy:.2f}%")
```

```
1800/1800 [==============================] - 4s 2ms/step
Validation Set: MSE (Best Model): 221968.46333784197
Validation Set: R² (Best Model): 0.05460865856011232
Validation Set: Mean Absolute Percentage Error (MAPE) (Best Model): 0.1916726313102403
Validation Set: Prediction Accuracy (Best Model): 80.83%
1800/1800 [==============================] - 3s 2ms/step
Test Set: MSE (Best Model): 223100.2293186439
Test Set: R² (Best Model): -0.11547708457787853
Test Set: Mean Absolute Percentage Error (MAPE): 0.15894765366549096
Test Set: Prediction Accuracy: 84.11%
```

```python
y_pred_train = model.predict(X_train)
mape_train = mean_absolute_percentage_error(y_train, y_pred_train)
accuracy_train = (1 - mape_train) * 100  # Convert to percentage
print(f"Train Set: Mean Absolute Percentage Error (MAPE): {mape_train}")
print(f"Train Set: Prediction Accuracy: {accuracy_train:.2f}%")
```

```
7199/7199 [==============================] - 14s 2ms/step
Train Set: Mean Absolute Percentage Error (MAPE): 0.23429281763180604
Train Set: Prediction Accuracy: 76.57%
```

Forecasting LSTM – Complex:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import tensorflow.keras as keras
from tensorflow.keras.callbacks import EarlyStopping
import random

# Set the random seed for reproducibility
random_seed = 42
np.random.seed(random_seed)
random.seed(random_seed)
tf.random.set_seed(random_seed)

# Define EarlyStopping callback to stop training if validation loss stops improving
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Build LSTM model
model_1 = Sequential()

# Adding the LSTM layer with 256 units
model_1.add(LSTM(units=256, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
# Adding a Dropout layer to prevent overfitting
model_1.add(Dropout(0.2))

# Adding another LSTM layer
model_1.add(LSTM(units=128, return_sequences=True))
model_1.add(Dropout(0.2))

# Adding another LSTM layer
model_1.add(LSTM(units=64, return_sequences=False))

# Adding a final Dense layer for output
model_1.add(Dense(units=1))  #Output layer for regression: predicting a single value (TSDM_MEAN)

# Compile the model
#opt = keras.optimizers.Adam(learning_rate=0.001) #1e-3
opt = keras.optimizers.Adam()
model_1.compile(optimizer=opt, loss='mse')  # Use mean squared error for the loss function

# Summary of the model
print(model_1.summary())

# Train the model
history_1 = model_1.fit(X_train, y_train,
          epochs=100,
          batch_size=32,
          validation_data=(X_val, y_val),
          callbacks=[early_stopping],
          verbose=2)
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 4, 256)            644096

 dropout_1 (Dropout)         (None, 4, 256)            0

 lstm_3 (LSTM)               (None, 4, 128)            197120

 dropout_2 (Dropout)         (None, 4, 128)            0

 lstm_4 (LSTM)               (None, 64)                49408

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 890,689
Trainable params: 890,689
Non-trainable params: 0
_________________________________________________________________
```