Crop Prediction with Time-series Data

1. Introduction

Background

Nature's most complex and unpredictable aspects are almost impossible to ignore. Whether sheltering inside your home during a cold winter or enjoying a warm summer on the beach, there is always the environmental factor, the factor driving most of our decisions. But, in the corporate world, these factors are yet as unpredictable and cumbersome as they are in our daily lives. In various industries, especially those in the primary sector, environmental factors are a major risk to production quantity and quality. Decisions made in these industries requires a fine touch of risk management and possibly years of experience to know how to react to the uncontrollable parameters, and even then to stumble at times.

Problem Statement

In the past decade, data-driven decisions became a major influence to drive success. Companies around the globe implement ways to collect as much as possible data to be able to analyse large datasets and derive valuable insights to optimise and manage their company effectively. However, the natural environmental's aspects can bring in a lot of noise that can defeat the purpose of an effective analysis and forecast.

The intent of time-series forecasting is to gain a deeper understanding of nature and how it reacts to the decisions we make everyday, both on personal and corporate level. The natural environment's endless complexity allows for freedom of exploration. This paper will focus specifically on the analysis of natural environments on corporate level. Here are some applications:

- Agriculture optimisation (e.g. crop yield, crop health and forestry)
- Power-generation optimisation (e.g. solar, wind, and water energy throughput)
- Public safety management (e.g. disaster preparedness)
- Construction management

This paper, however, will go into depth to predict crop yield and health.

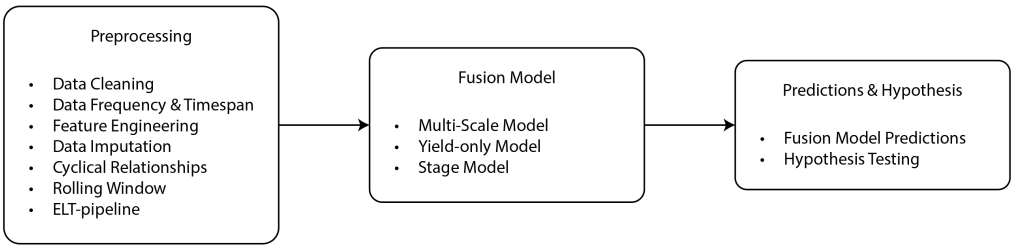
Objectives

This paper, titled "Crop Prediction with Time-series Data" is aimed at covering the following objectives:

- Strategies for data selection, noise reduction, data fusion and maintaining acceptable granularity throughout preprocessing.
- Imputing missing values by means of neighbouring methods.
- Describing how the ELT-pipeline was implemented.
- Methods to apply domain knowledge to a dataset. These methods include historical data, expert input, and heuristic models.
- The use of suitable forecasting models to train on various target variables.
- Ensure generalisations in the data is applicable and flexible.
- Handle when predictions becomes outliers.

Paper Structure

The paper will follow an sequential approach:



At each stage, the goal is to complete a different task (e.g., to clean data, fit a model, test a hypothesis), but the process remains sequential and this paper will follow a similar format.

The rest of the paper will be structured as follows: Section 2 briefly explains data selection, preprocessing strategies applied on the datasets. Lag and cyclical pattern management will also be discussed. Section 3 briefly looks at ways to enhance datasets through feature engineering and incorporating domain knowledge. A feedback-loop will also be discussed. Exploratory Data Analysis (EDA) will be done throughout Section 2 and 3. Section 4 implements forecasting models. Section 5 analyses and evaluates the hypotheses by means of testing and evaluating the models. Section 6 concludes by explaining how an ELT-pipeline was used.

2. Data Selection & Preprocessing

Data selection and preprocessing is an as-important factor as choosing an appropriate forecasting model. Data needs to be appropriate, and preprocessing needs to be done the right way to ensure the model understands relationships between data points in the dataset in order to derive insights.

Data Selection

Data selection is considered an integral part of being able to forecast accurately. Selecting inappropriate data leads to inaccurate prediction, as they have no correlation to the topic of interest. One might find that some inappropriate data might lead to some correlation, but in the long run might deviate from the topic of interest, as these two datasets have nothing in common (an example will be discussed later).

Data sets

An overview of all the base variables used and assessed in the study. Additional variables might be derived from these base variables later on (using feature engineering). The datasets containing these variables were generally well-maintained, requiring minimal data cleaning. The primary data preparation involved the application of consistent formatting, such as standardizing date formats.

Data Source	Variables	Location of Interest	Number of Variables	Frequency	Timespan	Number of records
OpenWeather	pressure, humidity, clouds, dew_point, rain, tempmax, tempmin, tempmean, tempdiurnal, solarradiation, solarenergy, uvi	Pretoria, South Africa	12	Daily	1979-2024	16696
GrainSA	Crop Yields (maize, wheat, groundnuts, sunflowerseed, sorghum, soybeans,	South Africa	n/a	Yearly (crop yields), Static (crop-specific knowledge)	1910-2023	Approx. 115 per crop

Data Source	Variables	Location of Interest	Number of Variables	Frequency	Timespan	Number of records
	barley, canola, oats), crop-specific knowledge (planting dates, crop stages)					

Comment

Picking local data proved to be an important factor when attempting to make accurate predictions, especially when considering cyclical patterns. The location of interest were considered to be Pretoria, South Africa, with fields located in and around the region (up to a maximum of approximately 50 kilometers).

Preprocessing Strategies

Data Frequency & Timespan

In the present study, the data sources contain different timespans and frequencies. In general, the aim is to preserve granularity. **Temporal alignment** is defined as the transformation $T(t)$ which maps t onto a common axis t'. This essentially means that the aim is to take various data sets and attempt to place them on the same timeline (also known as "data fusion"). This, however, is not always as straightforward as one might prefer. In the study, the result would contain missing values, which poses a problem of itself. There are industry standard methods to handle these.

- **Consistent formatting**

  It's not uncommon to see data sets that use different formatting, especially for date-specific variables. The outcome of formatting these variables needs to be concise, as it might misalign the data when interpreted incorrectly.

- **Data Imputation**

  Another method to handle missing values is to use similarity across the data set to predict values. The target row's known values is taken and then clustered with similar rows in the data set by using clustering methods such as **K-nearest neighbours** algorithm (also known as a KNN-algorithm). This method considers k neighbours and picks the neighbour that matches the criteria best.

- **Time Series Alignment**

  The problem with feature engineering and data imputation becomes a problem when the attempt is taken to fuse the smallest frequency (e.g. yearly data) into a larger frequency data set (such as monthly data). Yearly data simply lacks monthly resolution and might represent different values to different model outcomes. Possible methods are to make use of interpolation to estimate monthly data from yearly data by using seasonal patterns. The accuracy of the forecasting model depends heavily on the granularity of the data set (will be discussed in Section 5). As a last resort, the frequency of the time series data can be adjusted to a suitable range that would fit the majority of data sets, but is not recommended.

Cyclical Relationships and Lagged Data

Consider some arbitrary dates in data sets $D_1 = \{"2020-01", "2021-01", "2022-01", \ldots\}$ and $D_2 = \{"2020", "2021", "2022", \ldots\}$. Then it's presumable to convert D_1 to $\{"2021", "2022", "2023", \ldots\}$ such that $D_1 = D_2$, but for instance 2020's data could've been released in January 2020. This will cause D_1's data to be aligned 1 year behind D_2. This is called "lagged data".

The question arises whether data has a direct relationship (i.e. a strong correlation) or rather a lagged relationship, which might influence the outcomes of the model. Simply put, in natural environments, some variables are a consequence of other variables, which means there's a delay between cause and effect.

An example is shown below (Fig. 1) where rainfall directly influences the Normalised Difference Vegetation Index (NDVI) values. NDVI refers to the "greenness" of plants, and by intuition, we experience "greenness" in

nature as a result of good rain, but only after a while, since plants need to absorb water and grow.

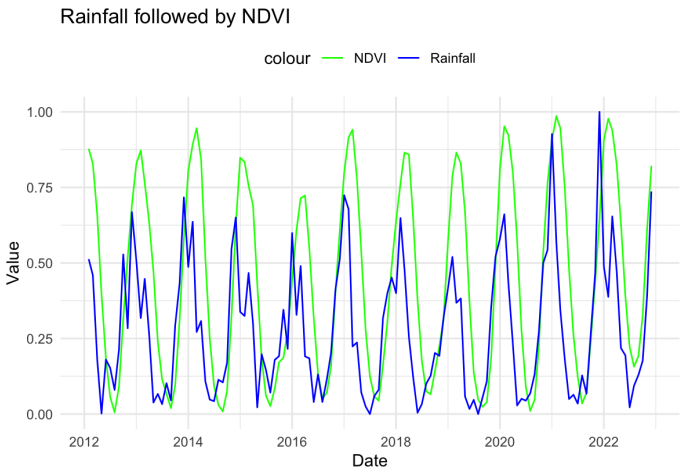


Figure 1: A plot showing the normalised relationship between rainfall and NDVI following seasonal patterns.

Similarly, one can observe a similar outcome with the effect of the average temperature on NDVI values (Figure 2). A lag is observed in July 2018 (red dot) where temperatures reaches a local minima and September 2018 (green dot) where NDVI values reaches a local minima. A two-month lag is observed in this instance.

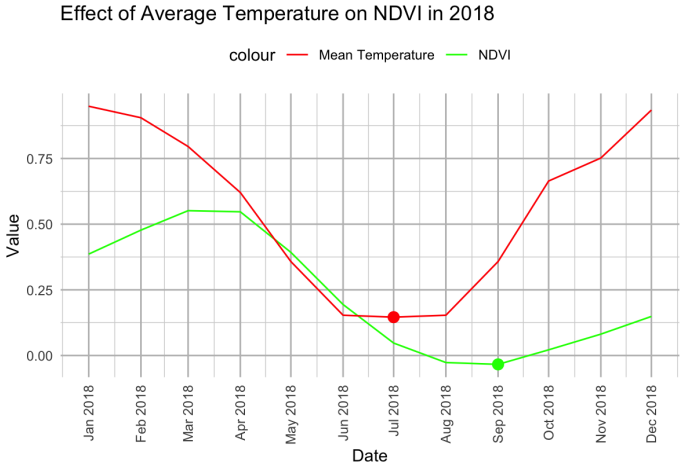


Figure 2: A plot showing the normalised relationship between the average temperature and NDVI in 2018.

So with all this in place, how can one effectively apply lag? And if so, does this affect the forecasting model? In this section of the paper, a strategic approach to cyclical patterns will be discussed, however in Section 5, these methods will be individually forecasted and in Section 6 these models will be opposed to one another in order to test Hypothesis 3.

Further investigation into lagged features is beyond the scope of this paper, as their inclusion would substantially increase the complexity of the analysis. While lagged variables can provide valuable insights, particularly in time-series forecasting, incorporating them requires additional preprocessing, parameter tuning, and model complexity that goes beyond the primary focus of this research.

Rolling Window

A rolling window is a certain area of interest is applied to a dataset. This is especially useful when working with environmental data where unpredictability comes into play. Rolling windows cater for this by taking the means of a certain window size (take n). The average is always calculated per window and stored in the last position of the window.

Let's say the current window is from a to b where $b - a = n$. Then, all the values from a to b is averaged and stored at the b^{th}-position. a and b is then incremented by one, retaining window size n. Consequently, the new value is then stored in the $(b + 1)^{th}$-position.

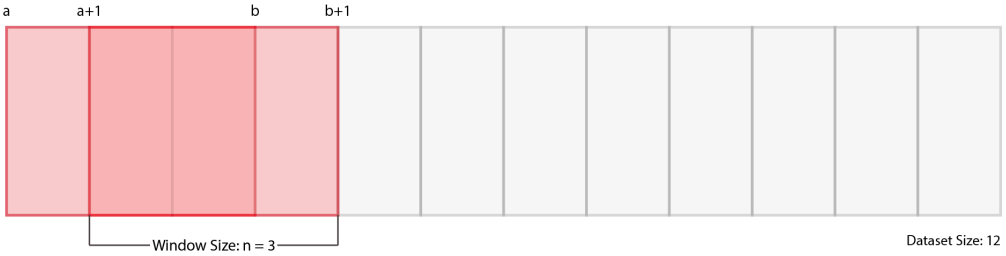


Figure 3: Visualisation of how a rolling window would typically be implemented

Further investigation into implementing a rolling window is beyond the scope of this paper, as it is already catered for mostly by the Multi-Stage Model (see Section 5).

Imputing Missing Values

Removing rows with missing values is a common method in the field of Data Science, but it comes with a major drawback that causes the size of the dataset to be reduced. Having limited data, considering that 100 years worth of data are only 100 rows in the dataset if working with yearly data, the need to be considerate before removing rows, as more data often provides better generalisation. But, with the help of a select set of algorithms, can rows with missing values be imputed rather than removed?

K-nearest neighbour (KNN)

K-nearest neighbours uses distance metrics to determine it's position. A KNN does not fill missing values with new values, instead it fills the missing values with the closest k entries (by taking the average in case of regression and the majority vote in case of classification) in the dataset that matches the entry with the missing value.

Scikit-learn provides a package called a `KNNImputer` which is explicitly designed to handle missing values in a dataset. The target is then predicted by interpolating the targets associated with the nearest neighbours in the dataset.

```
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=5)
imputed_data = data_to_impute.copy()
imputed_data[:] = imputer.fit_transform(data_to_impute)
```

3. Enhancing Datasets

Up and to this point, a sufficient amount of data got collected, accurately to the location of interest. These inputs serve as a baseline for feature engineering to take place on. Due to a lack of sensor inputs, soil moisture and temperature can't be scientifically calculated based on weather variables. Soil moisture and soil temperature does have an effect on plant growth (Russell, 1973), therefore, a dataset an unknown location was used to train a model on how to predict soil temperature and soil moisture on temperature and humidity.

Prediction of Soil Variables using XGBoost

For the prediction of soil variables, XGBoost has been used, as it works well with tabular data. More reasoning about why XGBoost was chosen as a general model to approach predictive data, see Section 4.

In this code, data has been split into 80%/20% train/test datasets. These were set on a seed value of 123, since they'll only be run once, and saved to serve as a consistent predictor for forthcoming data.

```
import xgboost as xgb
from sklearn.model_selection import train_test_split

X1 = data[['temperature', 'humidity']]
y1 = data['soil_moisture']

X2 = data[['temperature', 'humidity']]
y2 = data['soil_temperature']

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2, random_
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2, random_

xgb_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators = 10, seed = 1
```

For the best results, hyperparameter tuning with a cross-validation of 3 was used to accurately fit the model. Additionally, an iteration count of 50 was added to further improve training (although without great effect), totalling 150 fits.

```
random_search_1 = RandomizedSearchCV(
    xgb_model,
    param_distributions=param_dist,
    n_iter=50,  # Number of parameter settings that are sampled
    scoring='neg_mean_squared_error',
    cv=3,  # 3-fold cross-validation
    verbose=1,
    n_jobs=-1,
    random_state=42,
)

random_search_2 = RandomizedSearchCV(
    xgb_model,
    param_distributions=param_dist,
    n_iter=50,  # Number of parameter settings that are sampled
    scoring='neg_mean_squared_error',
    cv=3,  # 3-fold cross-validation
    verbose=1,
    n_jobs=-1,
    random_state=42,
)

# Fit the model
random_search_1.fit(X1_train, y1_train)
random_search_2.fit(X2_train, y2_train)
```

As a result, sufficient results were obtained as seen in Figure 4. The data that seemed to be outliers, got catered for during training to prevent overfitting. Complimentary to the graph, the Root Mean Square Error (RMSE) for soil moisture equated to 1.5507586655843835 and for soil temperature the RMSE equated to 0.07878415991080076. This is considered to be accurate, as soil moisture (expressed as an percentage) is expected be around ±1.55% from the actual value. Similarly, where the full dataset's temperature values were between 23.4ºC and 32.0ºC, a difference of ±0.07ºC is insignificant and can be considered an accurate predictor.

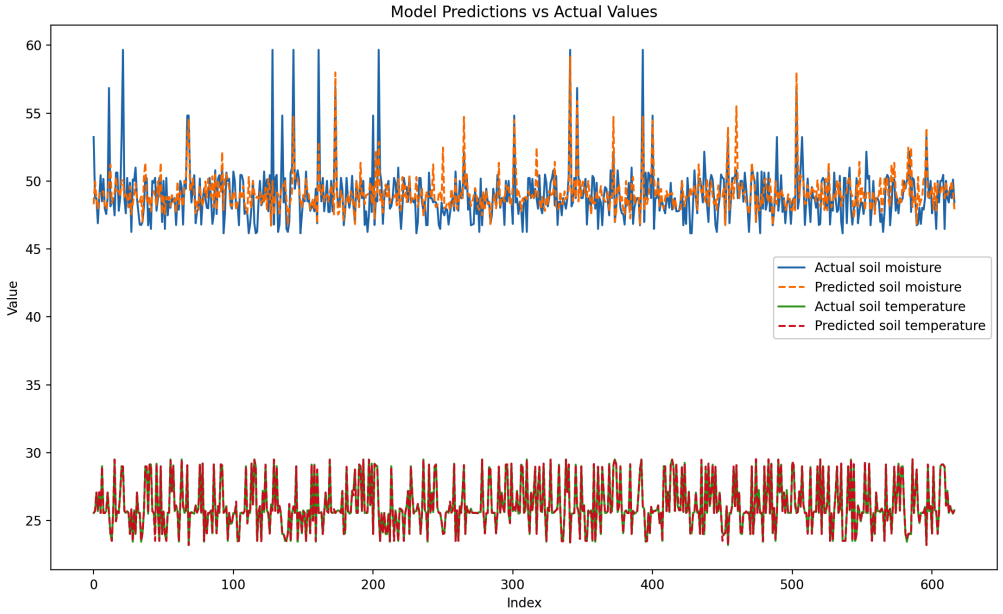


Figure 4: Soil-specific predictions using gradient boosting

Other features added

Additionally, a diurnal temperature has been added, a straight-forward calculation: $T_{max} - T_{min}$. This variable is significant for the model as "the difference (DIF) between day temperature (DT) and night temperature (NT) influences internode length, plant height, leaf orientation, shoot orientation, chlorophyll content, lateral branching and petiole and flower stalk elongation" (Jardar Myster, Roar Moe, 1995).

ELT-pipeline

Due to rapid development, an alternative approach has been taken to a regular ETL-pipeline. A regular ETL-pipeline extracts data from data sources, then transforms it by adding features, cleaning and fixing the dataset to a specific set of features. With an ELT-pipeline however, one can introduce features on the go as needed. For example, a new feature might be introduced that are more soil specific, or IoT integration might require the model to upgrade the historical dataset. Similarly, features can be rejected on the fly without affecting the original historical data stored in the database. With daily weather fetching, the use of an ELT-pipeline significantly improves server performance. All this makes an ELT-pipeline well-suited to evolving data requirements and dynamic use cases.

4. Fusion Model

As mentioned earlier, different frequencies might have a different outcome on various variables. Rainfall, for example, might affect crop growth in the long run. Some crop growth stages might focus on specific weather variables throughout the stage, possibly different than a possible preceding or following stage. Rather than trying to fit the best frequency for each crop type, the approach to an ensemble model ended up to be a reasonable approach. With the ensemble model, the two polar opposites, and everything in between, are catered for. One extreme notices short-term changes in weather, while the other extreme allows for long-term patterns to be recognised.

XGBoost

Reasoning about the model needs to be adequate, and serves a great importance to the outcome of the predictions. XGBoost (Extreme Gradient Boosting) in itself forms part of the ensemble learning category, and more specifically the gradient boosting aspect. Gradient boosting is a boosting algorithm which trains the model sequentially, trying to improve on the previous iteration. Gradient boosting algorithms consist of multiple decision trees where each tree makes a prediction, and finally, per definition, averages all the trees'

predictions. The idea of gradient boosting, as opposed to random forest, is to combine weak models into a single strong model, but also by following an iterative approach trying to improve upon the preceding model.

Regardless of claims and theory, throughout hypothesis testing, XGBoost has proven to be a sufficient choice for predicting crop yield.

Yield-only Model

Weather variables might not always correctly predict the yield correctly. This is simply because other environmental and non-environmental variables also affect the outcome. According to experts in the field, technological advancements hugely affects yield throughput (see Figure 5). For this reason, the year of production has been added as an feature.

As a baseline model, the Yield-only Model was used as an unbiased estimator for crop prediction. This is important as it will serve as the alternative hypothesis in Section 5. The results, again, were sufficient to be included as part of the Fusion Model with a good RMSE of 0.08906415374372288.

ARIMA, another popular forecasting method, has been used. As seen in Figure 5, the results were rather disappointing, with a RMSE of 2.750881415364275, a value considered insufficient. As a result, ARIMA were excluded to form part of the Fusion Model.

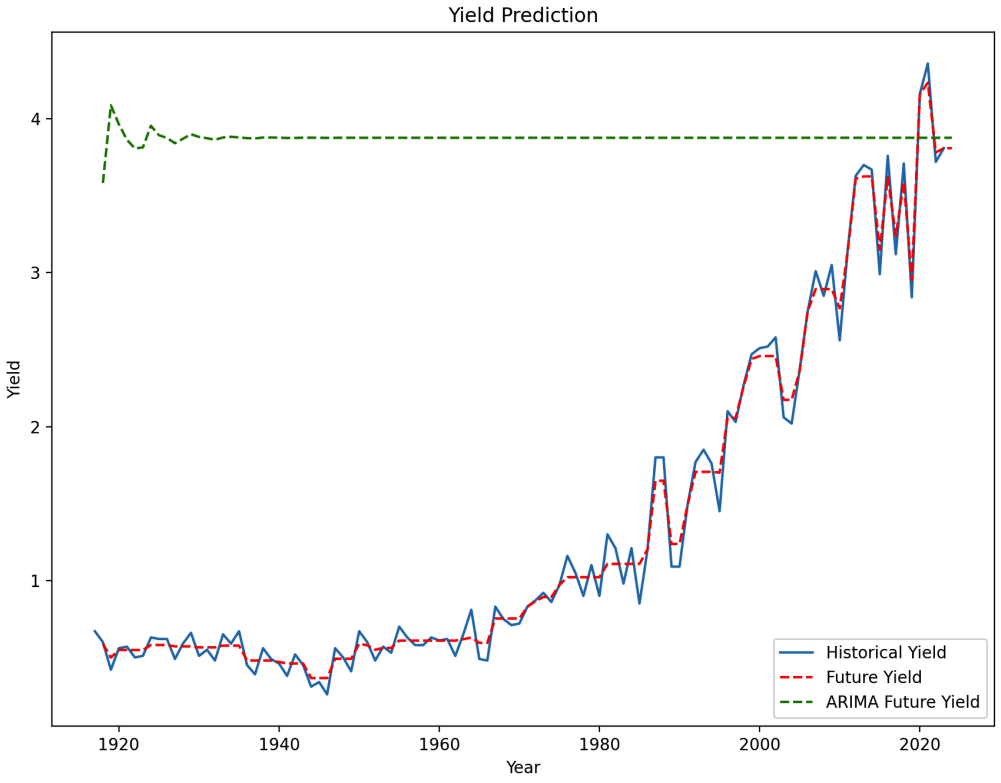


Figure 5: Difference between XGBoost and ARIMA to predict yield using only yield as feature

Multi-Scale Model

With a bit of thought, the attempt to counter the rolling-window problem, a intuitive approach has been taken to divide the dataset into several frequencies, with hopes to search for specific patterns in a certain frequency. Why does this counter the rolling-window problem? More specifically, does this take into account features that require time to highlight meaningful insights? The reason this works so well is because it takes into account a wide range of frequencies. Granularity, as mentioned in Section 2, is a consideration, since the variable of interest is daily predictions. If it's a bad weather day, does it affect the yield? Not necessarily. This is also part of the power of XGBoost not to overreact on small changes. With a multi-scale model, overfitting is virtually eliminated, as possible overfitting is cancelled out by a model with lower frequency (e.g. yearly).

The Multi-Scale model works by creating a separate dataset for every frequency of interest (FOI), and then aggregating data into a required frequency. For example, when the FOI is quarterly, it is expected to have 4 entries per year, where each row is a result of the quarter's data. Sensible decisions has been made to sum certain variables and average others. A random variable is also added to prevent reproducibility and introduce some randomness.

Since the interest of predictions is specifically on the current date, values after the specified date is discarded for every year in the dataset. For example the date is 1 September 2024, then for every available year, the data after 1 September will be discarded. 2 September 1970 will then not be in the dataset. The reason behind this is for the simple fact that accumulated rainfall, for example, might not be significant at the start of a term in the FOI. So, when entering Quarter 3 of 2024, there might not be 200mm of rainfall. When rejecting values after the current date of interest, rainfall might be 10mm, which is much more realistic for fitting a suitable model.

Finally, each FOI is trained and used for independent predictions. A true ensemble approach is followed by averaging the outputs for each day predicted. The predictions obtained by each of the models were averaging around ±0.5 difference (RMSE) to the actual yield prediction (ton per hectare), which is sufficient for the use case. Due to performance limitations on the server, a XGBoost model with 1 iteration has been used, and is likely to have impacted the prediction accuracy. Finally, the predictions are returned to the Fusion Model.

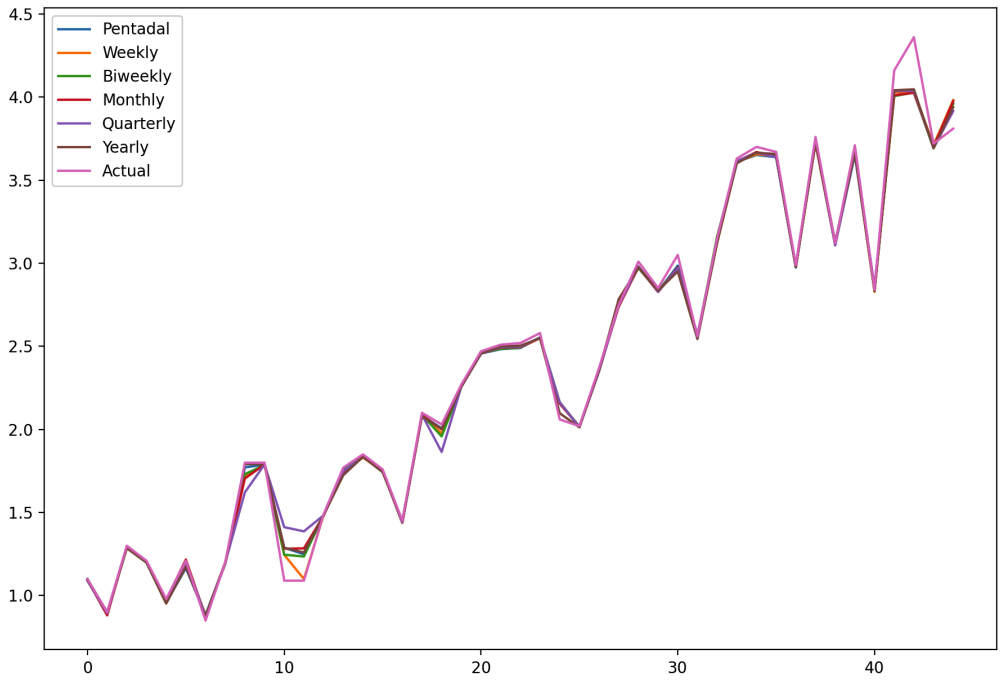


Figure 6: Result of the Multi-Scale model after fitting multiple models on the same dataset, but with different frequencies.

Stage Model

The Stage model works in a similar fashion to the Multi-Scale model. The Multi-Scale model uses the Gregorian calendar to identify sequences in the dataset. The Stage Model on the other hand uses crop heuristics such as planting dates and thereafter the growing stage of the crop. Data is then grouped together to form a single output per stage per year.

With the Stage model, the rows with dates after the current date is also discarded, similar to the Multi-Scale model. The same reason applies here where weak predictions were observed at the start of a stage, specifically with cumulative variables.

These crop heuristics are defined as part of the Crop class definition:

```
c = Crop(
    name="wheat",
    stages={
```

```
        "sowing": {"day": 111},
        "germination": {"day": 151},
        "tillering": {"day": 182},
        "heading": {"day": 243},
        "maturity": {"day": 304}
    },
    # other variables
)
```

The model showed good predictive ability with an RMSE of ±0.3 from the actual value. Finally, this is returned to the Fusion model where hypothesis testing takes place.

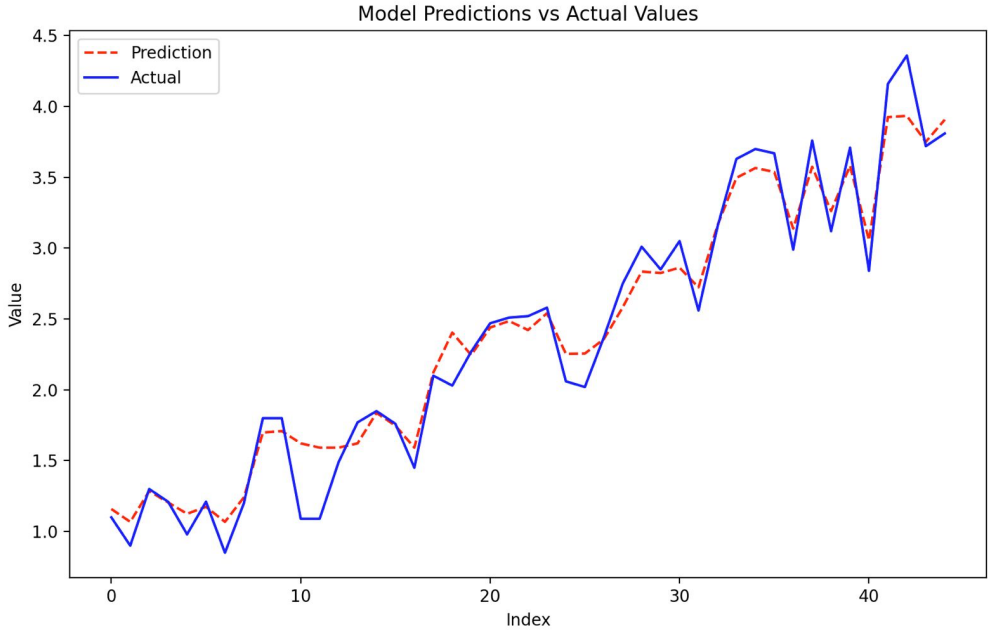


Figure 6: Result of the Multi-Scale model after fitting multiple models on the same dataset, but with different frequencies.

5. Hypothesis Testing

To ensure consistency in predictions, the model has been fitted with a standard hypothesis test that forms part of the pipeline that either rejects or retains the predictions made by the respective models. As a final result, an array of n requested days is returned.

```
msm_pred = self.msm.predict()
yom_pred = self.yom.predict()
sm_pred = self.sm.predict()[0] # if crop is in stage

# Yield-only Model
yom_last_5_years = yom_pred[-10:]
mean = np.mean(yom_last_5_years)

confidence_interval = 2.575 * np.std(yom_last_5_years) / np.sqrt(5)
ci_upper = mean + confidence_interval
ci_lower = mean - confidence_interval
final_predictions = []

for i in range(len(final_predictions)):
```

```
        # Test Multi-Scale Model
    if msm_predictions[i] > ci_upper or msm_predictions[i] < ci_lower:
            # Reject null hypothesis
        if sm_pred is not None and (sm_pred <= ci_upper and sm_pred >= ci_lower):
                # Check if the crop is in stage and test StageModel
            final_predictions[i] = sm_pred
        else:
                # Reject null hypothesis, and store alternative hypothesis (Yield-Only
            final_predictions[i] = mean
    else:
            # Do not reject null hypothesis
        if sm_pred is not None:
                # Check if the crop is in stage
            final_predictions[i] = (msm_predictions[i] + sm_pred) / 2
        else:
                # Else store the average.
            final_predictions[i] = (msm_predictions[i] + mean) / 2
```

Conclusion

The fusion of models, supported by hypothesis testing, has demonstrated a versatile and adaptable framework for crop yield forecasting. Despite the constraints of limited data and time, the model has shown strong performance across a range of conditions. While the current model focuses on key stages of crop growth and frequency-based patterns, future improvements could explore additional variables, including environmental constants such as elevation, which may possibly affect crop yield.

The research conducted thus far provides a solid foundation for further investigation into more granular factors influencing yield, opening up possibilities for even more accurate and more region-specific predictions. With continued refinement and expansion, this model has the potential to make meaningful contributions to precision agriculture, specifically in third-world settings where expensive equipment is not viable. This can aid in enhancing food security and optimizing resource management in the face of changing climatic conditions.

Bibliography

- Russell, E. W. 1973. Soil conditions and plant growth. Longman Group Limited, London, Great Britian, 849 pp.
- Jardar Myster, Roar Moe, Effect of diurnal temperature alternations on plant morphology in some greenhouse crops—a mini review, Scientia Horticulturae, Volume 62, Issue 4, 1995, Pages 205-215, ISSN 0304-4238, https://doi.org/10.1016/0304-4238(95)00783-P.

https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.geeksforgeeks.org/ml-gradient-boosting/&ved=2ahUKEwjaydi86OOIAxV-hf0HHfU3PCEQFnoECB4QAw&usg=AOvVaw3ZJ685tS7WP3w-DPY160kE

https://www.nvidia.com/en-us/glossary/xgboost/

https://www.influxdata.com/time-series-forecasting-methods/

https://www.nobledesktop.com/classes-near-me/blog/time-series-analysis-environmental-applications

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10743348/

https://bookdown.org/igisc/EnvDataSci/time-series-visualization-analysis.html

https://www.youtube.com/watch?v=6mul7sWX2zI

https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html

https://www.youtube.com/watch?v=-aCF0_wfVwY&pp=ygUXdGltZSBzZXJpZXMgZm9yZWNhc3Rpbmc=

https://www.youtube.com/watch?v=cMcTwSAf9z0

https://www.yieldgap.org/atlas-advanced-users

https://www.yieldgap.org/web/guest/methods-soil-series

https://ndagis.nda.agric.za/portal/apps/webappviewer/index.html?id=8b72eb2a25c04660a1ab2b562f6ec0bf