

**“Development of a Dynamic, Data-Driven Reporting  
Module for the National Crop Yield Forecasting  
(NPCYF) System”**

---

# **Internship Report**

---

**INTERN: SHRITANU MONDAL**

**Course: M.A. Economics (Energy Economics), Banaras Hindu**

**Period of Internship: 25<sup>th</sup> August 2025 – 31<sup>st</sup> October 2025**

**Project Mentor Name: Samyabrata Roy  
IDEAS-TIH, ISI Kolkata**

**Report submitted to: IDEAS – Institute of Data  
Engineering, Analytics and Science Foundation, ISI  
Kolkata**

# ACKNOWLEDGEMENT

I wish to express my deepest gratitude to all the individuals and organizations who contributed to the successful completion of this project, "**Dynamic Reporting Module for Crop Yield Forecasts**," undertaken as part of the Autumn Internship Program on Data Science 2025.

A special note of appreciation goes to my mentor, **Mr. Samyabrata Roy** (IDEAS-TIH/ISI Kolkata), for his exceptional guidance, unwavering support, and valuable insights throughout every stage of this assignment. His technical expertise was crucial in shaping the **dynamic reporting framework** and resolving complex issues related to **PostgreSQL data mapping and Python docx implementation**.

I also extend my sincere thanks to the leadership and management team at the **Institute of Data Engineering, Analytics and Science Foundation (IDEAS-TIH)**, Indian Statistical Institute (ISI) Kolkata, including:

- **Dr. Sanghamitra Bandyopadhyay**, Director, IDEAS-TIH
- **Dr. Agnimitra Biswas**, Chief Executive Officer (CEO)
- **Ms. Bidisha Dobe**, Operations & Training Manager
- **Mr. Diptendu Dutta**, Technology Lead

The team provided the excellent training infrastructure necessary for this internship experience. As an aspiring data professional, this opportunity has offered valuable hands-on exposure to real-world projects focused on **centralized data storage, integration, report automation**, and **dynamically adapting to database updates**.

Thank you once again to everyone at IDEAS-TIH for making this internship a truly valuable and meaningful learning experience.

**Regards,**

**SHRITANU MONDAL**

# **Abstract**

The National Program for Crop Yield Forecasting (NPCYF), spearheaded by the Mahalanobis National Crop Forecast Centre (MNCFC) under the Ministry of Agriculture and Farmers Welfare, plays a vital role in India's agricultural planning. As part of the FASAL (Forecasting Agricultural output using Space, Agro-meteorology, and Land-based observations) initiative, it integrates remote sensing, meteorological, and ground-based data to generate crop yield forecasts across states and districts. However, manual report preparation and static data management often delay insights and limit scalability. This project, **“Development of a Dynamic, Data-Driven Reporting Module for the National Crop Yield Forecasting (NPCYF) System,”** focuses on developing a dynamic and automated reporting framework that streamlines data retrieval, processing, and document generation. Using PostgreSQL for centralized data storage and Python for data integration, transformation, and report automation, the system generates state-wise yield forecast reports in Word format, dynamically adapting to database updates. The framework enhances usability, reduces human intervention, and ensures consistency and transparency in reporting. This data-driven approach contributes to improving operational efficiency, accuracy, and timeliness in India's crop yield forecasting ecosystem.

key keywords:

1. **Dynamic Reporting** (Covers the solution's main feature: flexibility and automation.)
2. **Crop Yield Forecasting** (Covers the core domain and application.)
3. **NPCYF / FASAL** (The specific project environment/platform.)
4. **PostgreSQL / Python** (The core technology stack used for implementation.)
5. **Data Automation** (Covers the goal of streamlining data retrieval and document generation, focusing on efficiency.)

# Contents

<b>1. Introduction.....</b>	<b>5-8</b>
<b>1.1</b> Overview of the Organization .....	<b>5</b>
<b>1.2</b> Relevance and Purpose of the Project.....	<b>5</b>
1.3 Technology and Background Material Survey.....	5
1.4 Procedure Used.....	6
1.5 Week I: Foundational Tools and Concepts.....	7
1.6 Week II: Core Machine Learning and Web Integration.....	8
<b>2. Objectives .....</b>	<b>9-10</b>
<b>3. Methodology .....</b>	<b>11-14</b>
3.1 Tools and Technologies Used: .....	11
3.2 Data Collection and Data Handling: .....	11
3.3 Data Source and Schema: .....	12
3.4 Data Preparation Steps (Query and Extraction): .....	12
3.5 Step-by-Step Implementation of the Reporting Module: .....	13-14
<b>4. Data Analysis and Results: .....</b>	<b>15-16</b>
4.1 Descriptive Analysis of Data Structure: .....	15
4.2 Module Functional Results and Verification: .....	16
4.3 Result 1: Successful Dynamic Column Generation: .....	16
4.4 Result 2: Precise Report Formatting and Styling: .....	16
<b>5. Conclusion: .....</b>	<b>17</b>
<b>6. APPENDICES: .....</b>	<b>18</b>

# Introduction

## **Overview of the Organization:**

This project, undertaken as part of the Autumn Internship Program at the Institute of Data Engineering, Analytics and Science Foundation (IDEAS-TIH) at ISI Kolkata, focuses on developing a robust and flexible module for generating crop yield forecasts within the **National Program on Crop Yield Forecasting (NPCYF)** platform.

## **Relevance and Purpose of the Project:**

The core purpose of this project is to bridge the gap between predictive modeling outputs and decision-maker usability. The NPCYF platform relies on sophisticated Machine Learning (ML) and statistical models to forecast crop yields, which are vital for agricultural planning, policy formulation (by agencies like MoA&FW, GoI), and commodity market stabilization. The module's relevance is direct: it ensures that these complex, data-driven forecasts are instantly and accurately presented in a standardized, government-compliant report format (Word document). This automation eliminates manual data compilation, reduces human error, and allows the platform to provide **real-time, user-driven insights** based on user input (Season and Target Year).

## **Technology and Background Material Survey:**

The project is built around a modern data pipeline, integrating several key technologies. **PostgreSQL (Postgres)** serves as the authoritative source for data storage, holding historical crop data, state information, and model-generated yield forecasts and performance metrics (RMSE). **Python** forms the core of the backend logic, scripting the entire reporting module. Specifically, the **psycopg2** Python library facilitates secure and efficient database access for running complex SQL queries, and the **python-docx** library is used for Report Generation—structuring, styling, and populating the final required .docx format while adhering to specified layouts (e.g., single-column and multi-column designs).

The background material survey focused on two primary areas: understanding the **NPCYF data schema** (specifically tables for crops, states, years, methods, and yield values) and analyzing the **target report designs** (provided in the assignment attachments) to ensure the generated document precisely matches the required visual format, including header information, logo placement, and table structure.

## **Procedure Used:**

The implementation followed a structured, data-driven approach:

1. **Database Replication:** Constructed a minimalist PostgreSQL database structure with dummy data to mirror the NPCYF schema, allowing for local development and testing.
2. **Static Report Generation (Assignment 1):** Developed the initial Python script to connect to Postgres, fetch static crop and state data, and populated a basic Word document, verifying end-to-end data flow.
3. **Dynamic Column Implementation (Assignment 1.1):** Refactored the script to address the core problem statement:
  - **Dynamic Column Fetching:** Instead of hard-coding algorithm names (e.g., ARIMA, RF, XGBoost), the script was modified to dynamically query the available prediction methods (models) from the database for the given Target Year and create columns for them on the fly.
  - **Report Template Adaptation:** Implemented logic to support the different table designs specified by the user (single-column yield only and multi-column yield/RMSE) to generate reports that are adaptable to various display needs.
4. **Integration Framework:** The module was designed to be integrated into the NPCYF backend, accepting user inputs (Season and Target Year) to drive the dynamic data fetching and report structure.

## **Week I: Foundational Tools and Concepts (August 25 – August 31, 2025):**

<b>Day</b>	<b>Topic</b>	<b>Synopsis</b>
<b>Mon, 25 Aug</b>	Introduction - Welcome Note	Program orientation and setting expectations for the internship.
<b>Tue, 26 Aug</b>	Basic Statistics for Data Science	Covered essential statistical concepts, including measures of central tendency, variance, correlation, and probability distributions, which form the mathematical backbone of ML models.
<b>Wed, 27 Aug</b>	Data Visualization	Focused on techniques and tools for effective data presentation, emphasizing charts, graphs, and visual storytelling to convey insights from large datasets.
<b>Thu, 28 Aug</b>	Introduction to GitHub and Cloud Computing	Covered version control workflows using Git and GitHub (essential for collaborative development, including the project repository), and foundational concepts of cloud infrastructure.
<b>Sat, 30 Aug</b>	Streamlit 1 (widget handling)	Practical session on building interactive web applications using Streamlit, focusing on how to integrate and handle user input through various widgets.

**Week II: Core Machine Learning and Web Integration**  
**(September 1 – September 7, 2025):**

Day	Topic	Synopsis
Mon, 1 Sep	Stream lit 2 (database integration)	Advanced Streamlit session focused on connecting web apps to external databases (like Postgres) to fetch and display dynamic databases directly applicable to the NPCYF reporting module.
Tue, 2 Sep	Machine Learning 1 (regression)	Deep dive into regression models (Linear, Polynomial, etc.), focusing on predicting continuous output variables, which is the exact nature of crop yield forecasting.
Wed, 3 Sep	Machine Learning 2 (classification)	Covered classification algorithms (Logistic Regression, Decision Trees), focusing on predicting discrete categories, complementing the regression skills acquired.
Thu, 4 Sep	LLM Fundamentals	Introduced the core concepts, architectures, and use cases of Large Language Models (LLMs), including prompt engineering and potential applications in data analysis summarization.
Sat, 6 Sep	Communication Skills	Professional development session focused on presenting technical work clearly, documenting projects effectively, and collaborating within a team environment.

# Project Objective

The objective of this project is to design, develop, and implement a flexible, data-driven reporting module for the National Programme on Crop Yield Forecasting (NPCYF). This module is intended to serve as a crucial component of the NPCYF platform's backend, specifically handling the transformation of raw forecast data from the PostgreSQL database into formalized, user-friendly documents (Microsoft Word format). The main purpose is to automate the report generation process, making the outputs of complex Machine Learning models immediately actionable for agricultural policy makers and stakeholders.

## **What the Project is Trying to Illustrate**

This project is a practical demonstration of three core areas of modern data science and engineering: **Data Management**, **Business Logic Implementation**, and **Reporting/Visualization**.

1. **Illustrating Real-Time Data Utility:** We aim to illustrate how robust database design (PostgreSQL) is essential for supporting dynamic reporting needs. By designing the query logic to be driven by user input (Season and Target Year), the project shows how backend systems can turn static data stores into real-time, responsive information tools.
2. **Demonstrating Backend Flexibility:** The project highlights the importance of flexible code architecture. By dynamically fetching column headers (representing different forecasting algorithms like ARIMA, RF, XGBoost) directly from the database, the system is future-proof. It can automatically incorporate new machine learning models into the reports without requiring code modification, which is a key principle of scalable software design.
3. **Bridging Modeling and Policy:** The final report output in a familiar Word document format (DOCX) is an illustration of the necessary link between advanced statistical modeling and practical policy use. The project translates complex yield and performance metrics (like RMSE) into a structured, readable format required by governmental bodies like the Ministry of Agriculture and Farmers' Welfare (MoA&FW).

## **Key Project Objectives:**

The project is guided by the following core objectives:

- **Design a Dynamic Reporting Engine:** To create a Python-based module capable of accepting user-defined parameters (Season and Target Year) and autonomously generating a structured report document.
- **Establish Data Connectivity and Integrity:** To ensure secure and reliable connection to the PostgreSQL database, verifying that all fetched data—including state names, yield values, and performance metrics (RMSE) is accurate and complete.
- **Implement Dynamic Column Mapping:** To develop SQL and Python logic that dynamically identifies all available crop yield forecasting methods (e.g., ARIMA, RF, MoA&FW) in the database for the specified target year and builds the report tables with these methods as variable column headers.
- **Generate Standardized, Multi-Format Reports:** To successfully produce reports matching the two distinct visual formats required: the simpler Single-Column Yield Forecast report and the complex Multi-Column Report that includes both Yield and RMSE for each forecasting method.
- **Ensure Backend Integration Readiness:** To build the reporting module code in a clean, modular way that facilitates seamless integration with the existing NPCYF web frontend, enabling a smooth user experience from input to document download.

# Methodology

This section outlines the detailed methodology and step-by-step processes undertaken to develop the dynamic report generation module for the NPCYF platform. The work primarily involved **Software Development and Data Engineering**, focusing on reading data from a secure database, processing it, and formatting the output according to strict design specifications.

## Tools and Technologies Used:

The entire project was implemented using a combination of powerful open-source tools chosen for their stability and suitability for data-driven backend services:

- **Programming Language: Python** was chosen for its strong capabilities in data handling and document generation.
- **Database Management System (DBMS): PostgreSQL** was used as the backend database, specifically interacting with the tables that store crop, state, and forecast yield data.
- **Database Connector:** The **psycopg2** library (a PostgreSQL adapter for Python) was used to establish secure connections and execute all necessary SQL queries.
- **Document Generation:** The **python-docx** library was utilized to programmatically create, style, and populate Microsoft Word documents (.docx), ensuring precise adherence to the required report templates.
- **Version Control: Git and GitHub** were used for tracking all code changes, managing different versions of the Python script, and facilitating collaboration. The final code repository link is:  
[https://github.com/Shritanu23/IDEAS\\_Crop\\_Yield\\_Report\\_Forecast.git](https://github.com/Shritanu23/IDEAS_Crop_Yield_Report_Forecast.git)

## Data Collection and Data Handling:

It is important to clarify that this project did **not** involve primary data collection (like conducting a survey) or the development of new machine learning models. The methodology was centered on utilizing **Secondary Data** already residing within the NPCYF PostgreSQL system.

## **Data Source and Schema:**

The data required for report generation was collected exclusively from the NPCYF database. The key information was retrieved from the following conceptual tables:

1. **Crops Table:** Contains metadata like crop\_id and crop\_name (e.g., 'COTTON').
2. **States Table:** Contains metadata like state\_id and state\_name (e.g., 'Andhra Pradesh').
3. **Crop Yields Table (The Main Data Source):** This table is central. It links crop\_id, state\_id, year, **method** (the name of the forecasting algorithm, e.g., 'ARIMA', 'RF', 'MoA&FW'), **yield\_value**, and **RMSE** (Root Mean Square Error).

## **Data Preparation Steps (Query and Extraction):**

Since the project uses structured data directly from a relational database, traditional data cleaning or pre-processing steps like handling missing values or feature scaling were **not** required. The data preparation involved optimizing the SQL query process:

1. **Input Parameterization:** The script was designed to accept two critical inputs: **Target Year** and **Season** (though the current assignment focused only on Kharif).
2. **Targeted Query Construction:** A complex SELECT query was constructed to retrieve all necessary fields (state\_name, year, method, yield\_value, RMSE) for the specified crop (e.g., 'COTTON') and the relevant target and historical years.
3. **Data Structuring in Python:** The fetched tabular results were immediately structured into a nested Python dictionary format (data[crop][state][year][method] = value). This nested structure was crucial as it allowed for easy, dynamic lookup when populating the report table cells.

## **Step-by-Step Implementation of the Reporting Module:**

The project was executed in a phased manner, ensuring that core functionality was stable before implementing the dynamic requirements.

### **Phase 1: Environment Setup and Static Report Verification (Assignment 1 Foundation):**

1. **PostgreSQL Environment Setup:** Installed PostgreSQL locally and created the ShritanuDB.
2. **Schema and Dummy Data Creation:** Created the necessary tables (crops, states, crop\_yields) and inserted a minimal set of dummy data. This dummy data mimicked the format expected for the required years and methods (e.g., '2025-26 RF', '2024-25 MoA&FW').
3. **Initial Scripting:** Wrote a foundational Python script to connect to the database using psycopg2.
4. **Static Report Test:** The script was run to produce a basic, static report using hard-coded column names, successfully verifying that the end-to-end data flow (DB access to DOCX output) was functional.

### **Phase 2: Dynamic Data Implementation (Assignment 1.1 Core):**

This phase was the most critical, fulfilling the primary objective of creating a dynamic report that adapts to available forecasting methods.

1. **Dynamic Method Query:** The script was updated to execute a preliminary SQL query to find all **unique method names** (e.g., 'ARIMA', 'RF', 'XGBoost', 'MoA&FW') that exist in the database for the given Target Year and Crop.
2. **Header Generation:** The unique methods found in step 1 were combined with the required historical data headers (e.g., '2024-25 MoA&FW') to create a definitive, ordered list of all columns for the report table. This list is generated dynamically every time the script runs.
3. **Optimized Data Fetching:** A refined main query was executed to fetch the yield\_value and RMSE for every state across *all* dynamically identified methods and years.

4. **Report Structure Mapping:** The python-docx library was used to construct the Word document table row by row. For each state, the script iterates through the dynamically generated column list (the headers) and correctly maps the fetched data from the Python dictionary structure into the corresponding cell. If a specific state/year/method combination has no data, the cell is left blank, ensuring visual accuracy.

### **Phase 3: Formatting and Finalization:**

1. **Header and Footer Customization:** Custom functions were implemented to add the project logo, title, subtitle, horizontal rules, and a dynamic footer with date and year information, precisely matching the provided visual templates.
2. **Table Styling:** Applied the 'Table Grid' style, set font sizes (Pt(10) for data, Pt(14) for main heading), and ensured correct cell alignment (left for State, center for Yield/RMSE values), including vertical alignment fixes using docx.xml for a professional appearance.
3. **Output Generation:** The final completed Word document was saved to the user-specified output path.

# Data Analysis and Results

Since the core objective of this project was the **functional development and integration of a dynamic reporting module** and not the development or statistical validation of new crop yield forecasting models, this section focuses on the **Descriptive Analysis of the Data Structure** and the **Verification of the Module's Functional Results**. No inferential analysis or hypothesis testing was performed, as the input data was secondary and provided by the NPCYF platform.

## Descriptive Analysis of Data Structure:

The analysis of the source data structure within the PostgreSQL database was critical, as it defined the necessary flexibility of the Python script.

Table 1: State-Wise Forecast Data (Cotton, Kharif 2025-26)

State	Year	Method	Yield Value (Kg/Ha)	RMSE
Andhra Pradesh	2025-26	Random Forest	359.47	25.12
Andhra Pradesh	2024-25	MoA&FW	418.00	52
Assam	2025-26	ARIMA	75.80	28.55
Assam	2024-25	MoA&FW	85.00	26
Gujarat	2025-26	Random Forest	450.11	22.09
Gujarat	2025-26	XGBoost	445.90	24.50

## Summary of Findings from Data Structure:

- Dynamic Methods:** The data clearly shows that for a single prediction year (2025-26) and a single crop (Cotton), there can be multiple forecasting Method entries (e.g., Random Forest, ARIMA, XGBoost). This confirmed the need for the module to **dynamically query and generate column headers** based on the available methods, which was the central challenge of the project.
- Historical Comparison:** The inclusion of official historical data (MoA&FW figures for 2024-25) requires the report to integrate both dynamic forecast columns and static historical comparison columns.
- Variable Metrics:** The presence of both Yield Value and RMSE confirmed the requirement to generate two distinct report types: the simple Single-Column report (Yield only) and the complex Multi-Column report (Yield + RMSE).

## Module Functional Results and Verification:

The success of the project is measured by the module's ability to accurately and dynamically generate the two required report formats.

### **Result 1: Successful Dynamic Column Generation:**

The primary outcome was the successful implementation of the dynamic column fetching mechanism.

- **Verification:** When running for the **Cotton, Kharif, 2025-26** scenario, the script successfully queried the database and dynamically produced the following sequence of column headers in the generated Word document, demonstrating flexibility.
  - **Multi-Column Report Headers (Example):** State | 2025-26 Random Forest Yield | 2025-26 Random Forest RMSE | 2025-26 ARIMA Yield | 2025-26 ARIMA RMSE | 2024-25 MoA&FW Yield
- **Significance:** This result confirms that the module is future-proof; as new forecasting models are added to the database, the report framework will automatically incorporate them without requiring any changes to the Python source code.

### **Result 2: Precise Report Formatting and Styling:**

The module successfully replicated the precise visual specifications required by the NPCYF platform, including complex Word document formatting.

- **Verification:** The reports generated consistently matched the required templates, including:
  - Accurate placement and sizing of the ISI Logo.
  - Correct formatting of the main report title and header with a horizontal rule separator.
  - Applying the correct font sizes and text alignments (left for state names, centered for numerical yield values).
  - Successful implementation of the footer, including the dynamic generation of the current date and the hardcoded prediction year (2025-26).

# Conclusion:

The project successfully achieved its stated objective of designing and implementing a robust, flexible, and data-driven reporting module for the NPCYF platform. The methodology, focused on integrating Python, PostgreSQL, and the python-docx library, proved highly effective in meeting the complex requirements of generating standardized official reports.

## Justification of Conclusions

- **Flexibility is Validated:** The central finding is the successful implementation of **dynamic column mapping**. By retrieving forecasting methods directly from the database, the module is no longer reliant on hard-coded model names. This justifies the conclusion that the system is **scalable and future-proof**, capable of handling new forecasting algorithms without maintenance overhead.
- **Integration is Confirmed:** The modular design of the Python script, which accepts user input parameters (Target Year, Season) through command-line arguments (simulating frontend inputs), confirms its readiness for seamless integration into the NPCYF backend framework, thereby streamlining the user query-to-report download process.
- **Compliance is Achieved:** The ability to accurately replicate both the Single-Column and Multi-Column report formats, including precise header, footer, and styling elements, justifies the conclusion that the generated output meets the high standards of official compliance required by governmental end-users.

## Recommendations for Future Work

Based on the development process, the following recommendations are proposed to enhance the module's utility and integration:

1. **Frontend API Integration:** Convert the standalone Python script into a robust backend API service (e.g., using Python Flask or Django) that can receive HTTP requests from the NPCYF frontend and return the generated .docx file as a response stream.
2. **Report Caching Mechanism:** Implement a caching layer to store generated reports temporarily. If a user requests the exact same report parameters (Year, Season, Crop) within a short time frame, the cached file can be served instantly, reducing database load and generation time.
3. **PDF Output Option:** Extend the python-docx functionality to include an option for exporting the final report directly as a PDF, which is often a required format for official documentation.

# APPENDICES

## **References:**

- **PostgreSQL Official Documentation:** For SQL querying, data types, and connection details.
- **psycopg2 Documentation:** For Python-PostgreSQL connection and handling database cursors.
- **python-docx Documentation:** For programmatic creation, manipulation, and styling of Microsoft Word documents, including using OxmlElement for advanced styling like vertical cell alignment.
- **NPCYF Project Specifications:** Internal documents and templates defining the required report layouts and data fields.
- MNCFC – Mahalanobis National Crop Forecast Centre, Government of India.
- FASAL Project – <https://www.ncfc.gov.in/fasal>
- IDEAS-TIH, ISI Kolkata – Internship Training Material
- OpenAI Python-docx Documentation – <https://python-docx.readthedocs.io>
- PostgreSQL Documentation – <https://www.postgresql.org/docs>