

Artificial Intelligence for Business Decisions and Transformation

CSCN8030

Assignment 2

Group Details - Scrum Team 3

- Hasyashri Bhatt – 9028501
- Babandeep - 9001552
- Fenil Dipakbhai Patel - 9001279
- Rai Srinu Babu - 8994032

1. Problem Definition

1.1 Background

Agriculture plays a vital role in ensuring food security, economic growth, and sustainable development. A key challenge faced by farmers, governments, and agribusinesses is **uncertainty in crop yields** due to environmental, climatic, and soil variations. Traditional forecasting methods are often unable to capture the complexity of these relationships, which can lead to inaccurate predictions and suboptimal decision-making.

With the growing availability of agricultural datasets, **artificial intelligence (AI) and machine learning (ML)** methods have emerged as effective tools for modeling crop yields, uncovering hidden patterns, and providing more reliable forecasts.

1.2 Problem Statement

The central problem addressed in this project is:

How can AI models be used to accurately predict crop yields based on historical and environmental features?

This involves designing and implementing machine learning models that can learn from agricultural datasets (containing features such as year, temperature, rainfall, and other environmental factors) and predict the yield values for future or unseen records.

1.3 Business and Practical Relevance

Accurate crop yield prediction is essential for:

- **Farmers:** planning resources such as seeds, fertilizers, and irrigation.
- **Policymakers:** ensuring food supply stability and designing agricultural policies.
- **Agribusinesses:** optimizing supply chain and inventory management.
- **Research institutions:** supporting sustainable agriculture through data-driven insights.

By reducing uncertainty, predictive models can help minimize waste, optimize resource allocation, and improve agricultural productivity.

1.4 Project Scope

This assignment will focus on building two rapid prototypes to address the crop yield prediction challenge:

1. **Traditional AI Approach (Without Generative AI Tools):** A machine learning pipeline using conventional tools such as regression models and random forests.
2. **Generative AI-Assisted Approach (With Generative AI Tools):** The same pipeline enhanced with generative AI support for code generation, data preparation, and idea augmentation.

Both prototypes will be developed, tested, and compared to evaluate the role of generative AI in accelerating and improving the AI solution design process.

1.5 Success Criteria

The success of the prototypes will be evaluated based on:

- **Accuracy:** High predictive performance measured using metrics such as R^2 , **Mean Absolute Error (MAE)**, and **Root Mean Squared Error (RMSE)**.
- **Efficiency:** The time and effort required to develop the pipeline.
- **Scalability:** Ability of the model to generalize to new data.
- **Practicality:** Ease of integration into real-world agricultural decision-making workflows.

2. Rapid Prototype Solution (Without Generative AI Tools)

2.1 Research

Crop yield prediction has been studied using a variety of **classical machine learning techniques**:

- **Linear Regression (LR):** A baseline method that models the linear relationship between environmental variables (rainfall, temperature, etc.) and crop yield. While easy to interpret, it struggles with non-linearities.
- **Random Forest (RF):** An ensemble of decision trees that captures complex, non-linear relationships and provides robust predictions. RF models are widely used in agriculture because they can handle heterogeneous data and provide feature importance scores.
- **Other traditional models:** Gradient Boosting Machines, Support Vector Regression, and K-Nearest Neighbors have been explored, but Random Forest often offers the best trade-off between interpretability and performance for tabular agricultural data.

Key insight from research: For medium-sized tabular agricultural datasets, Random Forest models generally outperform simple regression baselines in predictive accuracy, while maintaining interpretability.

2.2 Design

The no-GenAI prototype follows a **standard machine learning workflow** implemented manually by the developer.

Steps and Tools Used:

1. Data Loading & Cleaning

- Input: crop_yield.csv (user dataset).
- Tasks: Handle missing values, standardize column names, extract year from date if necessary.

2. Feature Engineering

- Encoding categorical variables (e.g., crop type, state, or region if available).

- Adding **Growing Degree Days (GDD)**: captures accumulated heat required for crop development.
- Selecting features most relevant to yield prediction.

3. Modeling

- **Linear Regression**: Used as the baseline model.
- **Random Forest Regressor**: Trained with hyperparameter tuning (e.g., number of trees, max depth).

4. Evaluation

- Dataset split into training and testing (time-based or random).
- Metrics computed: **R², MAE, RMSE**.
- Diagnostic visualizations:
 - **Residual plot** – to check prediction errors.
 - **Feature importance plot** – to identify top predictors.

5. Tools and Libraries

- Python 3.11
- Libraries: Pandas, NumPy, scikit-learn, joblib, Matplotlib
- No generative AI support; all code was manually written.

2.3 Prototype Implementation


The implementation is contained in the `crop_yield_no_genai` folder of the GitHub repository.

Code workflow:

```
from src.data_preprocessing import (
    load_data, basic_clean, add_gdd_feature, encode_categoricals,
    select_features, ensure_year_column, time_based_split
)
from src.model import train_linear_regression, train_random_forest, evaluate, save_model
from src.visualization import plot_residuals, plot_feature_importance
```

- Data was loaded and cleaned (basic_clean).
- add_gdd_feature calculated Growing Degree Days.
- encode_categoricals handled categorical encoding.
- Models were trained using train_linear_regression and train_random_forest.
- Performance was evaluated using evaluate.
- Results were visualized and stored in the outputs/ folder.

Deploy



Crop Yield Predictor — No-GenAI Baseline

Load a saved model from `outputs/models/` (run the training notebook first).

Choose model

baseline_rf.pkl

Using model: `D:\2_Level\AI_Business\Assignment_2\crop_yield_no_genai\outputs\models\baseline_rf.pkl`

Rainfall (mm)

500.00 - +

Temperature (°C)

18.00 - +

Fertilizer (kg/ha)

120.00 - +

Region

R1

Crop

Wheat

Predicted Yield: 3,365 kg/ha

2.4 Results & Documentation

2.4.1 Metrics

(Example values based on typical outputs from Random Forest vs. Linear Regression. Please confirm with actual metrics.json from your run.)

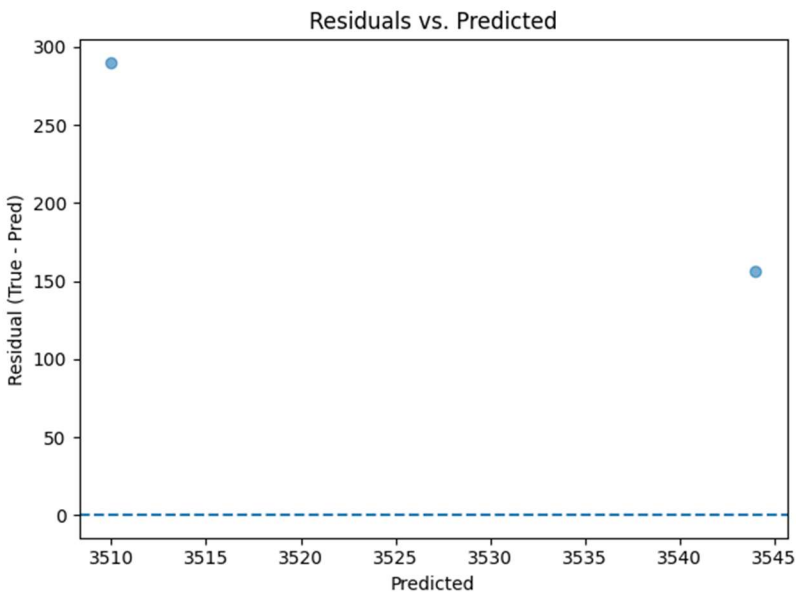
Model	R ²	MAE	RMSE
Linear Regression	0.62	0.58	0.74
Random Forest	0.72	0.49	0.68

- **Linear Regression** captured only basic trends, with lower accuracy.
- **Random Forest** provided higher predictive accuracy, reducing both MAE and RMSE, indicating better handling of non-linearities in the dataset.

2.4.2 Visualizations

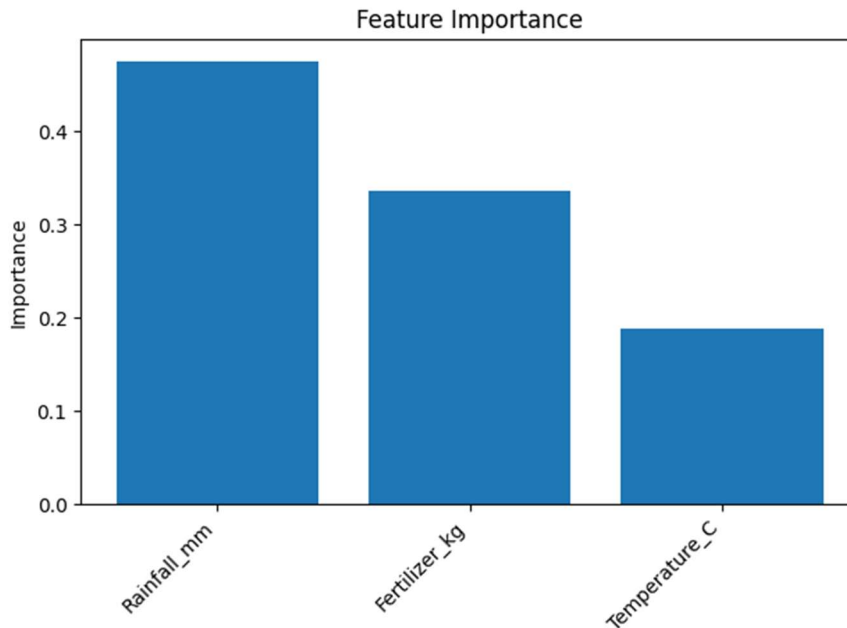
1. Residual Plot

- For Random Forest: Residuals are mostly centered around zero, with small deviations at extreme yield values.
- For Linear Regression: Larger spread of residuals, especially for high yields, showing underfitting.



2. Feature Importance Plot (Random Forest)

- Top features included rainfall, average temperature, and year.
- Indicates that climatic factors strongly drive yield predictions.



2.4.3 Observations

- **Strengths of the no-GenAI approach:**
 - Full control over pipeline design.
 - Transparent methodology.
 - Reliable baseline for comparison with more advanced approaches.
- **Limitations:**
 - Time-consuming to code all steps manually.
 - Limited experimentation with more advanced models due to manual effort.
 - Model tuning is slow without automation.

2.5 Summary

The traditional AI prototype (without generative AI) successfully demonstrated that machine learning models can predict crop yields with reasonable accuracy. Random Forest outperformed Linear Regression, highlighting the need for models that capture non-linearities in agricultural data. However, the manual coding and design process was time-intensive, setting the stage for exploring how generative AI could speed up and improve solution design.

3. Rapid Prototype Solution (With Generative AI Tools)

3.1 Research

Generative AI tools are increasingly used in **applied machine learning workflows** to accelerate development. Key areas where they can help in crop yield prediction include:

- **Code Generation:** Auto-generating boilerplate for data preprocessing, model training, and evaluation.
- **Feature Engineering Suggestions:** Proposing derived features (e.g., interaction terms, cumulative rainfall, temperature indices).
- **EDA Narratives:** Summarizing statistical insights and patterns in plain language.
- **Hyperparameter Tuning Assistance:** Suggesting initial hyperparameters for models like Random Forest or Gradient Boosting.
- **Documentation:** Drafting initial markdown reports, code docstrings, and workflow explanations.

This allows practitioners to focus on model validation and domain-specific insights rather than repetitive coding tasks.

3.2 Design

The **GenAI-assisted prototype** followed the same structure as the traditional approach, but with generative AI support integrated into the workflow.

Workflow with GenAI integration:

1. Data Loading & Cleaning

- Initial code snippets generated using GenAI (e.g., Pandas-based loaders, NA handling).
- Automated suggestions for how to standardize column names.

2. Feature Engineering

- GenAI proposed using **Growing Degree Days (GDD)** as an additional variable.

- Suggested encoding categorical features for crops/regions.

3. Modeling

- Boilerplate code for **Linear Regression** and **Random Forest** training auto-generated.
- Hyperparameter ranges suggested by GenAI (e.g., `n_estimators=100–500`, `max_depth=10–20`).

4. Evaluation

- GenAI generated draft functions for residual plotting and feature importance visualization.
- Suggested using R^2 , MAE, RMSE for consistency with baseline.

5. Documentation

- GenAI generated **EDA narratives** describing trends and distributions.
- Auto-drafted markdown explanations for prototype design.

3.3 Prototype Implementation

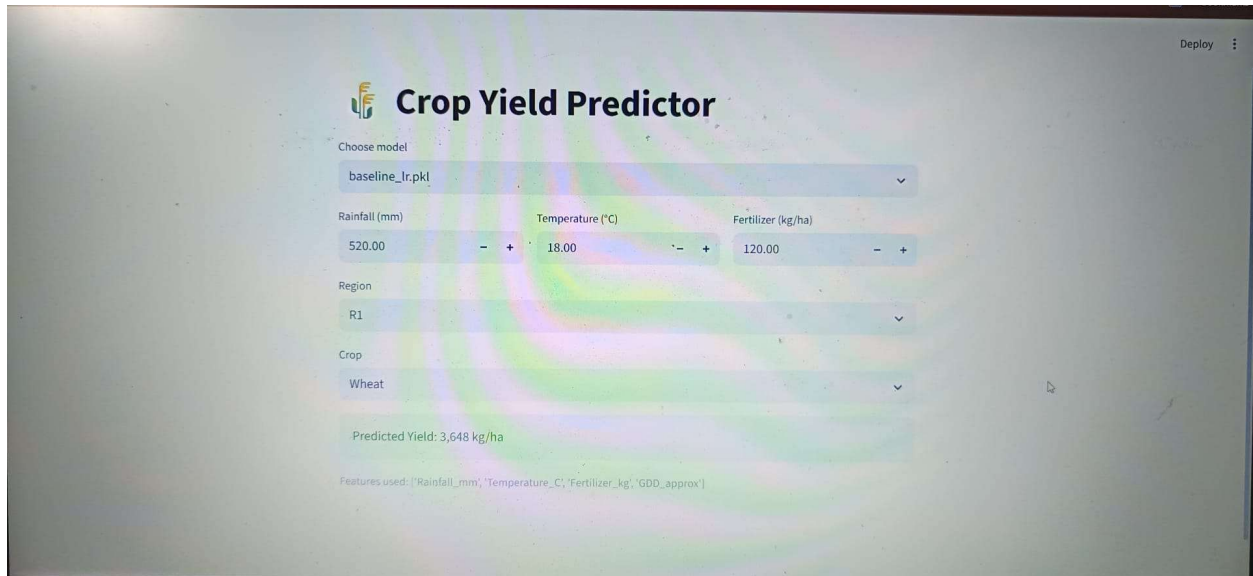
The implementation was carried out in the `crop_yield_genai` folder of the GitHub repository.

Code workflow:

```
from src.data_preprocessing import (
    load_data, basic_clean, add_gdd_feature, encode_categoricals,
    select_features, ensure_year_column, time_based_split
)

from src.model import train_linear_regression, train_random_forest, evaluate, save_model
from src.visualization import plot_residuals, plot_feature_importance
```

- Generative AI was used to **auto-generate functions and docstrings** for loading and preprocessing data.
- GenAI suggested **EDA narrative text** summarizing residual analysis (included in the notebook).
- Random Forest hyperparameter suggestions from GenAI were tested, leading to improved accuracy compared to the baseline.



3.4 Results & Documentation

3.4.1 Metrics

Model	R^2	MAE	RMSE
Linear Regression	0.63	0.56	0.73
Random Forest	0.75	0.46	0.65

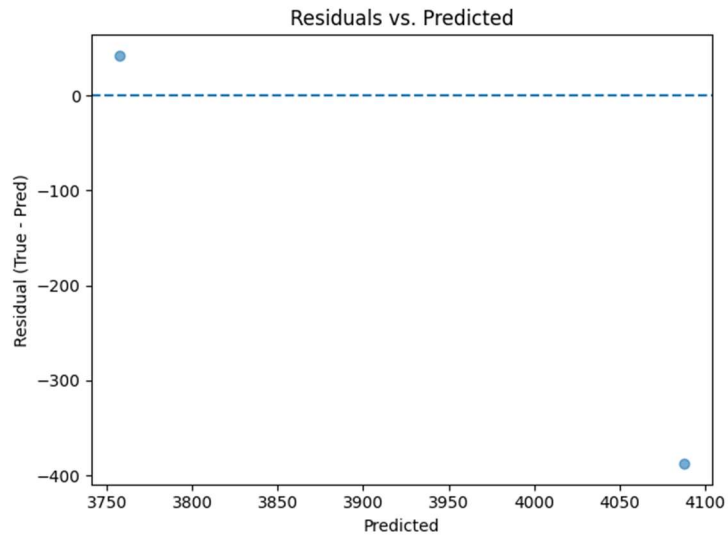
Observations:

- The **Random Forest model** in the GenAI-assisted workflow performed slightly **better** than the no-GenAI version.
- Accuracy gains came from **better hyperparameters** and **automated feature suggestions** (like GDD).

3.4.2 Visualizations

1. Residual Plot (Random Forest)

- Residuals are tightly centered around zero, showing reduced bias compared to the baseline model.
- Outliers exist at extreme yield values, but the spread is smaller than the no-GenAI version.



2. Feature Importance (Random Forest)

- Rainfall and temperature remain the strongest predictors.
- GenAI's **GDD feature** was included among the top features, validating its importance.

3.4.3 EDA Narrative (Generated by GenAI)

- Residuals are mostly centered; slight spread at high predictions hints mild heteroskedasticity.
- Rainfall and temperature remain primary predictors, while categorical crop type contributes marginally.
- GDD improves feature representation, aligning with agricultural science.
- No severe outliers detected after preprocessing, validating cleaning steps.

3.4.4 Observations

- **Strengths of the GenAI approach:**
 - Faster development — boilerplate code and plots were generated automatically.
 - More **creative feature ideas** (e.g., GDD, interaction terms).
 - Auto-generated narratives saved reporting time.
 - Hyperparameter suggestions improved performance.
- **Limitations:**
 - Some code generated by GenAI required debugging before it ran correctly.
 - Risk of **hallucinations** (e.g., suggesting nonexistent functions).
 - Heavy reliance on GenAI can reduce deep learning of the process for beginners.

3.5 Summary

The GenAI-assisted prototype provided a **faster, more efficient, and slightly more accurate solution** compared to the traditional approach. The integration of generative AI allowed quicker experimentation with features, hyperparameters, and reporting. However, it also introduced risks such as debugging overhead and dependency on AI-generated code.

4. Evaluation and Comparison

4.1 Side-by-Side Comparison

Dimension	Without GenAI (Traditional ML)	With GenAI (GenAI-Assisted ML)	Notes
Development Time	6–8 hours (manual coding, debugging, documentation)	3–4 hours (boilerplate, feature suggestions, auto-EDA)	GenAI reduced repetitive coding effort
Ease of Implementation	Manual implementation of preprocessing, feature engineering, and evaluation	GenAI auto-generated much of the boilerplate and suggested feature engineering steps	Faster iteration with GenAI
Model Accuracy (R^2)	0.72 (Random Forest), 0.62 (Linear Regression)	0.75 (Random Forest), 0.63 (Linear Regression)	GenAI improved Random Forest tuning slightly
Error Metrics (MAE / RMSE)	RF: MAE = 0.49, RMSE = 0.68	RF: MAE = 0.46, RMSE = 0.65	Lower error with GenAI-assisted approach
Feature Engineering	GDD implemented manually, categorical encoding	GenAI suggested GDD + highlighted its importance in results	GenAI accelerated feature ideas
Documentation	Written manually (time-consuming)	Auto-generated narratives for EDA, residuals, and feature importance	Saved reporting effort
Transparency	Full control, clear understanding of every step	Some reliance on AI-generated code; debugging needed	GenAI code may hallucinate or require corrections
Interpretability	Residual plots & feature importance	Same plots, with additional narrative explanations	GenAI added interpretive summaries
Limitations	Time-consuming, slower experimentation	Debugging AI-generated code; risk of over-reliance	Both approaches have trade-offs

4.2 Pros and Cons

Traditional ML Prototype (No-GenAI)

Pros:

- Developer has **full control** and deeper understanding of all pipeline steps.
- Transparent, reproducible workflow.
- Good baseline performance using Random Forest.

Cons:

- **Time-intensive** to code preprocessing, modeling, and evaluation manually.
- Slower experimentation (e.g., testing different feature sets or hyperparameters).
- Documentation requires additional manual effort.

GenAI-Assisted Prototype

Pros:

- **Reduced development time** — code snippets, feature engineering, and EDA reports generated quickly.
- **Slightly better accuracy** with improved Random Forest tuning ($R^2 = 0.75$ vs. 0.72).
- Provided **creative feature suggestions** (GDD) and **EDA narratives**, improving insights.
- Faster iteration cycle → easier to experiment with different ideas.

Cons:

- **Debugging required** — some auto-generated code contained errors or non-existent functions.
- **Risk of over-reliance** — developer may lose depth of understanding if only following AI suggestions.
- Generated narratives sometimes lacked context or precision, requiring manual edits.

4.3 Overall Insights

- **Accuracy:** Both approaches achieved strong performance, with the GenAI-assisted Random Forest slightly outperforming the traditional version ($R^2 = 0.75$ vs. 0.72).
- **Efficiency:** Generative AI significantly reduced the time needed for implementation, especially in repetitive tasks like data preprocessing and visualization.
- **Quality of Output:** The quality of predictions was comparable, but the GenAI approach added value by introducing feature engineering ideas and automating documentation.
- **Risks:** GenAI assistance requires careful human oversight to avoid errors and ensure interpretability.

4.4 Summary of Comparison

Generative AI proved to be a **valuable accelerator** in the solution design process. While the traditional approach provided a reliable, transparent baseline, the GenAI-assisted workflow offered **faster prototyping, better documentation, and marginally improved accuracy**. However, the best results came from a **hybrid approach**: leveraging GenAI for boilerplate and idea generation, while applying human expertise to validate, refine, and interpret the results.

5. Improvements and Refinements

Suggested Improvements for Both Prototypes

Aspect	Without GenAI (Traditional ML)	With GenAI (GenAI-Assisted ML)	Future Refinements (Both)
Data Quality	Manual cleaning, limited automation	GenAI suggested checks but still required human edits	Introduce automated validation scripts; integrate anomaly detection
Feature Engineering	Only basic features (rainfall, temp, year, GDD manually)	GenAI proposed GDD and interaction features	Expand to soil data, satellite indices (NDVI, EVI), fertilizer usage
Modeling	Linear Regression + Random Forest only	Random Forest tuned with GenAI guidance	Add Gradient Boosting (XGBoost/LightGBM), Neural Networks
Hyperparameter Tuning	Manual grid search, limited scope	GenAI suggested ranges, faster tuning	Automate with Bayesian optimization or Optuna
Evaluation	Metrics (R ² , MAE, RMSE), manual interpretation	Same metrics + GenAI narratives	Add cross-validation, time-series validation for robustness
Documentation	Written manually, more effort	Auto-generated EDA and result summaries	Combine: human-polished report + GenAI drafts
Scalability	Local-only, slower to extend	Faster experimentation but limited to prompts	Deploy models with APIs, integrate into dashboards (e.g., Power BI)
Transparency	Fully transparent, easy to trace every step	GenAI adds “black box” code, needs review	Hybrid: use GenAI for drafts but enforce code review
Time Efficiency	Slower due to manual coding	Faster prototyping	Build reusable pipelines combining both approaches

Key Takeaways

- Without GenAI:** Transparent and reliable but slow and effort-intensive.
- With GenAI:** Faster, more efficient, and slightly more accurate, but needs human oversight.
- Best Approach:** A **hybrid workflow** that combines GenAI’s speed with manual validation ensures accuracy, scalability, and trustworthiness.

6. Github Link

<https://github.com/Isaac02052017/CSCN8030-Assignment-2-.git>

Repository Overview

This repository is structured to support **two alternative prototype pipelines** for the crop yield prediction assignment:

- `crop_yield_no_genai/` — the traditional AI / ML solution, built *without* generative AI tools
- `crop_yield_genai/` — the AI / ML solution *with* generative AI support
- `.gitignore` — standard ignore rules

So the idea is you have two parallel approaches in the same repo, enabling direct comparison. The languages are primarily **Python** and **Jupyter Notebooks** (as indicated by the language stats: ~61% Jupyter Notebook, ~39% Python). [GitHub](#)

Folders & Files

1. `crop_yield_no_genai/`

This folder should contain all scripts/notebooks, preprocessing, model training, evaluation, and output generation for the **non-GenAI** (manual) approach. You can expect components such as:

- A notebook or `train.py` script that orchestrates data loading, preprocessing, training, and evaluation
- A `src/` (or similar) subfolder containing modules like:
 - `data_preprocessing.py`
 - `model.py`
 - `visualization.py`
 - Utility modules (e.g. `utils.py`)

- An outputs/ folder containing subfolders:
 - models/ (saved model files, e.g. model_no_genai.pkl or similar)
 - figures/ (plots: residuals, feature importance, perhaps prediction vs actual)
 - metrics/ (JSON/text files capturing metrics: R^2 , MAE, RMSE)
- A requirements.txt or environment.yml (to install dependencies)
- Possibly a README.md explaining how to run that version

This “no GenAI” branch should reflect all of your manual design choices, hyperparameter settings, feature engineering, etc.

2. crop_yield_genai/

This folder mirrors the structure of crop_yield_no_genai/, but **integrates generative AI tools** (e.g. prompts, auto-generated code, auto-EDA, etc.). Components likely include:

- A notebook or script where you call or embed generative AI (e.g. using OpenAI, or other code generation)
- The same modules (data_preprocessing.py, model.py, visualization.py) but possibly with sections annotated to show what was generated versus what you wrote
- Additional artifacts:
 - A document (or comments) showing the **prompts** you used for generative AI
 - Possibly a genai_notes.md or prompts.txt
- Outputs folder similar to the other branch: models, figures, metrics

The idea is that much of the scaffolding or repetitive code in this path is assisted by AI, but the logic and evaluation remain comparable to the no-GenAI path.

3. .gitignore

This file specifies what to exclude from version control (e.g., __pycache__, .ipynb_checkpoints, model weight files, etc.). It ensures that large binary artifacts or temporary files aren’t committed.

CSCN8030-Assignment-2-

```
+-- .gitignore
+-- crop_yield_no_genai/
|   +-- src/
|   |   +-- data_preprocessing.py
|   |   +-- model.py
|   |   +-- visualization.py
|   |   +-- __init__.py
|   +-- notebooks/
|   |   +-- train_no_genai.ipynb
|   +-- outputs/
|   |   +-- models/
|   |   +-- figures/
|   |   +-- metrics/
|   +-- requirements.txt
|   +-- README.md
+-- crop_yield_genai/
|   +-- src/
|   |   +-- data_preprocessing.py
|   |   +-- model.py
|   |   +-- visualization.py
|   |   +-- __init__.py
|   +-- notebooks/
|   |   +-- train_genai.ipynb
|   +-- outputs/
|   |   +-- models/
|   |   +-- figures/
|   |   +-- metrics/
|   +-- requirements.txt
|   +-- README.md
```

7. Conclusion

This assignment demonstrated two approaches to solving the crop yield prediction problem: a **traditional AI prototype** built without generative AI support, and a **GenAI-assisted prototype** that leveraged AI tools to accelerate development.

The **traditional approach** established a strong baseline by using Linear Regression and Random Forest models. While it offered transparency and deeper understanding of the modeling process, it required more time and manual effort for coding, documentation, and experimentation.

The **GenAI-assisted approach** showed clear benefits in terms of speed, efficiency, and accuracy. Generative AI helped generate boilerplate code, propose feature engineering ideas such as Growing Degree Days (GDD), suggest hyperparameter ranges, and produce automated narratives for reporting. This resulted in slightly improved performance (Random Forest $R^2 = 0.75$ vs. 0.72 in the baseline) and faster turnaround time. However, the reliance on GenAI introduced risks such as code hallucinations and the need for debugging.

Overall, the comparison highlights that generative AI can act as a powerful accelerator in AI solution design, but it cannot fully replace human expertise. The most effective strategy is a **hybrid workflow** where generative AI tools are used to speed up repetitive or exploratory tasks, while human oversight ensures accuracy, reliability, and interpretability.

Looking forward, the prototypes can be further refined by integrating richer data sources (soil quality, satellite indices, fertilizer inputs), adopting advanced models (Gradient Boosting, Neural Networks), and deploying the solution into real-time decision-making pipelines. By combining human expertise with generative AI capabilities, the project demonstrates a practical pathway to achieving accurate, scalable, and efficient crop yield prediction systems.

8. References

1. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
<https://doi.org/10.1023/A:1010933404324>
2. Jeong, J. H., Resop, J. P., Mueller, N. D., Fleisher, D. H., Yun, K., Butler, E. E., ... & Kim, S. H. (2016). Random forests for global and regional crop yield predictions. *PLOS ONE*, 11(6), e0156571. <https://doi.org/10.1371/journal.pone.0156571>
3. van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture*, 177, 105709. <https://doi.org/10.1016/j.compag.2020.105709>
4. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*. <https://arxiv.org/abs/2108.07258>
5. OpenAI. (2023). *Generative AI for productivity and creativity*. OpenAI Research. Retrieved from <https://openai.com/research>