Project code Description

The Python code for predicting crop yield and pest infestation in wheat using a combination of Convolutional Neural Networks (CNN) and Random Forest models. Here’s a breakdown of its key components, insights, and suggestions for improvement:

**1. Data Loading and Preprocessing – Sai Vamsi**

* The dataset is loaded from a CSV file, with preprocessing steps including loading images for pest prediction, normalizing them, and predicting pest infestation levels via a CNN model. Predictions are then added as a new feature in the dataset.
* **Insight**: By treating pest infestation as a feature for yield prediction, the model can learn how pest levels impact yield.
* **Suggestion**: Validate that all image paths are valid and that images are accessible. Additionally, consider resizing images to a lower dimension (e.g., 100x100) if memory or processing speed is a concern.

**2. Yield Prediction with Random Forest - Reshmi**

* The script uses a Random Forest Regressor to predict crop yield based on multiple features, applying a Grid Search for hyperparameter optimization.
* **Insight**: Random Forest is a strong choice for regression tasks involving structured data due to its robustness to overfitting and interpretability.
* **Suggestion**: Since Random Forest models can be computationally intensive, consider reducing the grid search parameters to speed up training or using RandomizedSearchCV to explore a broader range of parameters with less computation.

**3. Cross-Validation and Evaluation – Krithika & Ramesh**

* Both yield and pest models are evaluated using cross-validation and error metrics like MSE, RMSE, MAPE, and explained variance. The pipeline includes combining yield predictions with pest predictions for adjusted yield estimates.
* **Insight**: The combination of yield and pest predictions as a weighted metric can provide a more accurate estimate of crop health and output.
* **Suggestion**: Consider using different weighting schemes (perhaps dynamically tuned during training) to optimize the adjusted yield predictions further. Additionally, add feature standardization to ensure features are on similar scales.

**4. Image Processing and Pest Detection with CNN – Shovona and Ziya**

* The CNN model is trained on image data to detect pest infestation, which is then used as an input for yield prediction.
* **Insight**: The CNN architecture is appropriate for binary classification but could be improved with data augmentation techniques to prevent overfitting and improve generalization.
* **Suggestion**: To enhance model performance, use a deeper architecture (e.g., adding more Conv2D layers or increasing the dense layer neurons). Use more complex augmentations, such as horizontal flips, rotations, and zooms, to improve model robustness.

**5. Final Evaluation and Visualization – Akash & Ujwal**

* The script includes thorough metrics for model evaluation and various visualizations such as feature importance, prediction vs. actual plots, and residual plots.
* **Insight**: Visualization of feature importance is particularly valuable in interpreting the model, especially for Random Forest.
* **Suggestion**: Save visualizations to files for later analysis and implement early stopping during CNN training to prevent overfitting.

**6. General Improvements**

* **Error Handling**: Add try-except blocks to handle errors, especially in image loading and model training, to ensure robustness.
* **Scalability**: If the dataset grows, consider parallelizing certain parts of the code (e.g., batch processing images) to improve processing efficiency.
* **Model Storage**: Save trained models (using model.save() for CNN and joblib for Random Forest) for reusability and version control.

This code is close to being functional, but it likely requires a few adjustments to run without errors, even with a correct dataset. Here are some considerations and potential issues to check:

**1. CNN Model Initialization and Usage - Shovona and Ziya**

* The model variable, which refers to the CNN for pest prediction, is defined after the function predict\_pest\_infestation is called in the code. To use model for pest prediction, the CNN needs to be defined and trained before the function is called.
* **Fix**: Move the CNN model building, compiling, and training steps above the predict\_pest\_infestation function definition, or load a pretrained model if already available.

**2. Image Path and Dataset Assumptions - Sai Vamsi**

* The code assumes that each entry in data['image\_path'] points to a valid image file. If any path is invalid or if image\_path values aren’t actual file paths, the code will fail when trying to load images.
* **Fix**: Ensure image\_path contains valid file paths to images. Add error handling in predict\_pest\_infestation to manage missing or unreadable images gracefully.

**3. Data Structure and Features - Sai Vamsi**

* The code drops ['yield', 'pest\_infestation', 'image\_path'] columns in X, assuming these are present in the dataset. It also assumes that other features required by the model are available in data.
* **Fix**: Verify that the dataset contains all the necessary features, including 'yield' and 'pest\_infestation'. If using different feature names, modify the column names accordingly.

**4. Grid Search Parameters - Reshmi**

* The grid search for the Random Forest model may take considerable time, depending on the dataset size and computational power. This process could timeout or become resource-intensive if running on limited hardware.
* **Fix**: Start with a smaller grid, or use RandomizedSearchCV to explore a subset of parameter combinations initially.

**5. Cross-Validation Scoring for Regression - Krithika & Ramesh**

* The scoring method used is 'neg\_mean\_squared\_error', which should work fine, but keep in mind that MSE scoring will produce negative values, as higher negative values indicate lower MSE.
* **Suggestion**: Ensure the interpretation of cross-validation scores aligns with this convention (e.g., absolute values when displaying scores).

**6. Adjusted Yield Prediction Calculation - Shovona and Ziya**

* The calculation of optimum\_output uses pest\_predictions directly, assuming it represents the impact on yield in a straightforward manner.
* **Suggestion**: Adjust pest impact calculation based on the actual relation between pest infestation and yield (e.g., calibrate pest prediction to a percentage scale if needed).

**7. Data Generators for Image Preprocessing - Sai Vamsi**

* The paths to train\_dir and validation\_dir must contain correctly structured subdirectories (e.g., each class in its folder). If this structure is not adhered to, flow\_from\_directory will raise errors.
* **Fix**: Ensure the directories are structured correctly, with separate subfolders for each class if binary classification is used.

**8. Final Evaluations and Plotting - Akash & Ujwal**

* Plotting and residual analysis should work but may produce unexpected results if there are discrepancies in dataset distribution or data quality.
* **Suggestion**: Verify the dataset for outliers or inconsistencies that could skew metrics like MSE or explained variance, leading to unexpected plot patterns.