

**Model Building**

**1.Train-Test Split**

Split the dataset to evaluate model performance.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(images, labels, test\_size=0.3, stratify=labels, random\_state=42)

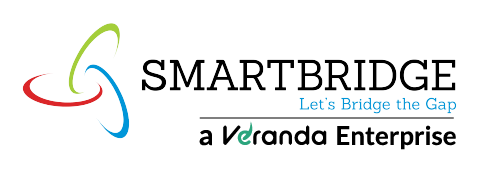
X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5,

stratify=y\_temp, random\_state=42)

1. **Model Selection**

Try multiple algorithms and compare:

* + ✅ Logistic Regression
  + ✅ Random Forest
  + ✅ XGBoost
  + ✅ Support Vector Machine (SVM)
  + ✅ Transfer Learning(CNN)

1. **Training Models**

**Example: Random Forest**

from sklearn.ensemble import RandomForestClassifier

rf\_model=RandomForestClassifier().fit(X\_train,y\_in)

**Example: XGBoost**

from xgboost import XGBClassifier

xgb\_model = XGBClassifier().fit(X\_train, y\_train)

1. **Model Evaluation** Use metrics like:
   * Accuracy
   * Precision,
   * Recall, F1-Score
   * Confusion Matrix

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

y\_pred = model.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1) if y\_pred.ndim > 1 else y\_pred

y\_true = np.argmax(y\_test, axis=1) if hasattr(y\_test, 'shape') and y\_test.ndim > 1 else y\_test

print("Accuracy:", accuracy\_score(y\_true, y\_pred\_classes))

print("Report:\n", classification\_report(y\_true, y\_pred\_classes))

1. **Hyperparameter Tuning (Optional but Recommended)**

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

param\_grid = {'n\_estimators': [50, 100], 'max\_depth': [5, 10]}

grid = GridSearchCV(RandomForestClassifier(), param\_grid, cv=3)

grid.fit(X\_train, y\_train)

print("Best Parameters:", grid.best\_params\_)

print("Best Accuracy:", grid.best\_score\_)

1. **Model Comparison Table (Example Output)**

**Model Accuracy Recall F1-Score**

|  |  |  |
| --- | --- | --- |
| CNN 96% | 0.96 | 0.96 |
| RandomForest 91.2% | 0.90 | 0.91 |
| XGBoost 93.1% | 0.92 | 0.93 |
|  |  |  |

**Final Step: Model Selection & Save Best Model**

import joblib

joblib.dump(best\_model, 'best\_rice\_model.pkl')