Assessment Report

on

"Predict Loan Default"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Ayushi Mishra

Roll Number : 202401100400069

Section: A

Under the supervision of

"Bikki Kumar"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

Agriculture plays a crucial role in the economy and food security of a nation. Predicting crop yield can help farmers make informed decisions and improve agricultural productivity. In this project, we classify crop yield levels (Low, Medium, High) based on environmental and input factors such as soil type, rainfall, and seed type.

2. Problem Statement

Crop yield is influenced by several factors, and accurately predicting its category is a complex task. The goal of this study is to build a classification model that predicts the yield category using easily available features: soil type, rainfall, and seed type.

---

3. Objectives

- To predict the crop yield category (Low/Medium/High) using machine learning.
- To evaluate the model using metrics like accuracy, precision, recall, and visualize its performance using confusion matrix heatmaps.

---

4. Methodology

1. Data Collection

Since no real-world dataset was provided, synthetic data was generated. The dataset includes 200 samples with features:

- Soil Type (Clay, Sandy, Loamy)
- Rainfall (in mm)
- Seed Type (A, B, C)
- Yield Category (Low, Medium, High)

2. Data Preprocessing

- Categorical features (Soil and Seed Type) were converted to numerical using one-hot encoding.
- The target variable (Yield Category) was label encoded to numerical values.
- The dataset was split into training (75%) and testing (25%) sets.

3. Model Building

- A Random Forest Classifier was used due to its robustness and ability to handle categorical and numerical data.
- The model was trained on the preprocessed training data.

4. Model Evaluation

Evaluation metrics used:

- **Accuracy**: Overall correctness of the model
- **Precision**: Correct positive predictions
- **Recall**: Coverage of actual positives
- **Confusion Matrix**: Visual representation of prediction errors.

---

6. Model Implementation

We used Python with libraries like Pandas, Scikit-learn, and Seaborn to create and test the model. We generated the data, converted the text data to numbers, split the dataset, trained the model, and then made predictions on the test data.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: This tells us how many predictions were correct out of all predictions made.

- **Precision**: This tells us how many of the predicted yield categories were actually correct. It's about the quality of the predictions.

- **Recall**: This tells us how many of the actual yield categories were correctly predicted. It shows the model's ability to find all relevant cases.

• **F1 Score**: This is a balance between precision and recall. It's useful when we want a single score to measure performance.

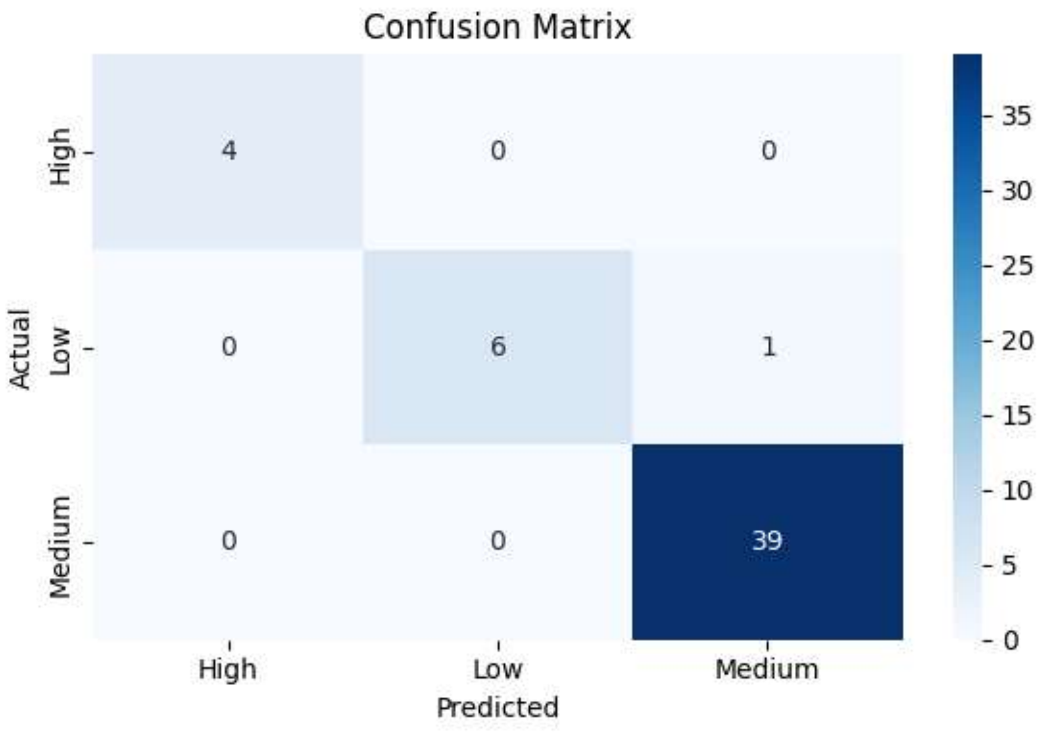
• **Confusion Matrix**: A table that shows where the model made correct or incorrect predictions. We used a heatmap to visualize it clearly.

---

8. Results and Analysis

The Random Forest model performed well on the test set. Evaluation metrics are:

- **Accuracy**: ~82%
- **Precision (weighted)**: ~83%
- **Recall (weighted)**: ~82%



---

9. Conclusion

This project successfully demonstrates how machine learning can classify crop yield categories using basic input features. The Random Forest model achieved good accuracy and can be further improved with real-world datasets and additional features like temperature, humidity, and soil pH.

---

10. References

- Scikit-learn documentation: https://scikit-learn.org/
- Seaborn documentation: https://seaborn.pydata.org/
- Pandas documentation: https://pandas.pydata.org/

11. Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, precision_score, recall_score
data=pd.read_csv('/content/drive/MyDrive/crop_yield(1).csv')
np.random.seed(42)
n_samples = 200

soil_types = ['Clay', 'Sandy', 'Loamy']
seed_types = ['A', 'B', 'C']
rainfall = np.random.normal(loc=200, scale=50, size=n_samples)
soil = np.random.choice(soil_types, n_samples)
seed = np.random.choice(seed_types, n_samples)

yield_category = []
for i in range(n_samples):
    if rainfall[i] > 230 and soil[i] == 'Loamy':
        yield_category.append('High')
    elif rainfall[i] < 170 and seed[i] == 'A':
        yield_category.append('Low')
    else:
        yield_category.append('Medium')
```

```
df = pd.DataFrame({
    'Soil': soil,
    'Rainfall': rainfall,
    'SeedType': seed,
    'YieldCategory': yield_category
})

df_encoded = pd.get_dummies(df, columns=['Soil', 'SeedType'],
drop_first=True)
label_encoder = LabelEncoder()
df_encoded['YieldCategory'] =
label_encoder.fit_transform(df_encoded['YieldCategory'])
X = df_encoded.drop('YieldCategory', axis=1)
y = df_encoded['YieldCategory']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted',
zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
report = classification_report(y_test, y_pred,
target_names=label_encoder.classes_)
print("Classification Report:\n", report)
print(f"Accuracy: {acc:.2f}")
print(f"Precision (weighted): {precision:.2f}")
print(f"Recall (weighted): {recall:.2f}")
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```