**Agriculture Data Analysis**

**Objective:** *Analyze agricultural data to improve crop yield, monitor soil health, and predict future farming trends.*

**📌 Step 1: Define the Problem Statement**

Some possible analysis topics:  
✅ **Crop Yield Prediction** (Predict the best crop based on soil & climate)  
✅ **Soil Quality & Fertility Analysis**  
✅ **Rainfall & Weather Impact on Crops**  
✅ **Market Price Trends for Agricultural Products**  
✅ **Pesticide & Fertilizer Usage Optimization**

For this guide, we will focus on **Crop Yield Prediction based on Soil & Weather Conditions**.

**📌 Step 2: Collect a Dataset**

You need a dataset that includes climate conditions, soil characteristics, and crop yield.

**🔹 Good Data Sources:**

* [**Kaggle - Crop Recommendation Dataset**](https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset)

**🔹 Dataset Features (Example)**

| **Feature** | **Description** |
| --- | --- |
| N (Nitrogen) | Nitrogen level in soil |
| P (Phosphorus) | Phosphorus level in soil |
| K (Potassium) | Potassium level in soil |
| Temperature | Temperature in Celsius |
| Humidity | Percentage of air humidity |
| pH | Soil pH value |
| Rainfall | Annual rainfall in mm |
| Crop | Recommended crop |

**📌 Step 3: Data Cleaning & Preprocessing**

**🔹 Load Data in Python**

python

CopyEdit

import pandas as pd

# Load the dataset

df = pd.read\_csv('crop\_recommendation.csv')

# Check basic information

df.info()

df.head()

**🔹 Handle Missing Values**

python

CopyEdit

# Check for missing values

print(df.isnull().sum())

# Fill missing values (example: replacing NaN with median values)

df.fillna(df.median(), inplace=True)

**🔹 Normalize Numerical Features**

python

CopyEdit

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df[['N', 'P', 'K', 'Temperature', 'Humidity', 'pH', 'Rainfall']] = scaler.fit\_transform(

df[['N', 'P', 'K', 'Temperature', 'Humidity', 'pH', 'Rainfall']]

)

**📌 Step 4: Exploratory Data Analysis (EDA)**

**🔹 Summary Statistics**

python

CopyEdit

print(df.describe())

**🔹 Correlation Heatmap**

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Feature Correlation Heatmap')

plt.show()

**🔹 Crop Distribution**

python

CopyEdit

plt.figure(figsize=(12,6))

sns.countplot(y=df['Crop'], order=df['Crop'].value\_counts().index, palette='viridis')

plt.title('Distribution of Crops in the Dataset')

plt.show()

**🔹 Impact of Rainfall on Crop Yield**

python

CopyEdit

sns.boxplot(x=df['Crop'], y=df['Rainfall'])

plt.xticks(rotation=90)

plt.title('Rainfall vs. Crop Type')

plt.show()

**📌 Step 5: Build a Machine Learning Model**

**🔹 Split Data into Train & Test Sets**

python

CopyEdit

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

X = df.drop(columns=['Crop']) # Features

y = df['Crop'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**🔹 Train a Classification Model**

python

CopyEdit

from sklearn.ensemble import RandomForestClassifier

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

# Initialize model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train model

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

# Make predictions

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

# Evaluate performance

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

print(classification\_report(y\_test, y\_pred))

**📌 Step 6: Model Optimization & Feature Importance**

**🔹 Hyperparameter Tuning**

Use **GridSearchCV** to find the best parameters.

python

CopyEdit

from sklearn.model\_selection import GridSearchCV

param\_grid = {'n\_estimators': [50, 100, 200], 'max\_depth': [None, 10, 20]}

grid\_search = GridSearchCV(RandomForestClassifier(), param\_grid, cv=5)

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

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

**🔹 Feature Importance**

python

CopyEdit

importances = model.feature\_importances\_

feature\_names = X.columns

plt.figure(figsize=(10,6))

sns.barplot(x=importances, y=feature\_names)

plt.title("Feature Importance")

plt.show()

**📌 Step 7: Data Visualization & Dashboard**

If you want to present insights in **Power BI or Tableau**, export the cleaned data.

python

CopyEdit

df.to\_csv('cleaned\_agriculture\_data.csv', index=False)

You can create:

* **Bar Charts**: Most recommended crops for different regions
* **Heatmaps**: Soil nutrient correlation
* **Pie Charts**: Distribution of crops

**📌 Step 8: Deploy Model (Optional)**

**🔹 Deploy as a Web App (Flask)**

python

CopyEdit

from flask import Flask, request, jsonify

import pickle

app = Flask(\_\_name\_\_)

# Load trained model

model = pickle.load(open('crop\_model.pkl', 'rb'))

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

prediction = model.predict([data['features']])

return jsonify({'Recommended Crop': prediction[0]})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**📌 Step 9: Summary of Requirements**

**📂 Tools Required:**

✅ Python (pandas, sklearn, seaborn, matplotlib)  
✅ Power BI / Tableau (for dashboards)  
✅ Flask / Streamlit (if deployment is needed)