Crop Recommendation System Using Machine Learning

"Data is the new soil. For me, it's like fertile soil that can yield new insights when cultivated with machine learning."
— Dave Waters

StreamLit ML App: https://crop-recommendation-model-noel-ninan.streamlit.app

GitHub Repo Link: https://github.com/NoelNinanSheri1307/CropRecommendationModel

Developed by: Noel Ninan Sheri
LinkedIn: https://www.linkedin.com/in/noel-ninan-sheri
GitHub: https://github.com/NoelNinanSheri1307
Dataset Source: Kaggle - Crop Recommendation Dataset

1. Introduction

The "Crop Recommendation System" is a machine learning-based web application designed to assist farmers and agricultural professionals in making informed decisions about the most suitable crop to cultivate based on real-time environmental factors.

Agriculture is highly dependent on climatic and soil parameters, and choosing the right crop can significantly impact yield, resource utilization, and sustainability. This project provides a smart and efficient way to leverage data science to improve decision-making in agriculture.

1.1 Need and Importance

- **Scientific crop selection** over traditional guesswork
- Enhanced **productivity** and **sustainability**
- Reduces **trial-and-error losses**
- Supports **precision agriculture**

2. Tech Stack Used

Frontend

- **Streamlit**: Lightweight, fast deployment of ML web apps
- **HTML/CSS (via Streamlit custom styling)**: For Times New Roman font and centered elements

Backend / ML

- **Python**: Core development language
- **Scikit-learn**: For model training, scaling, and encoding
- **Joblib**: For saving/loading models and objects
- **Pandas & NumPy**: For data manipulation

ML Techniques Used

- **Supervised Learning** (Classification)
- **StandardScaler** for feature normalization
- **Label Encoding** for categorical target transformation
- **Decision Tree Classifier / Random Forest (Depending on the model trained)**

3. Functional Modules

- **Input UI** for environmental parameters:
    - Soil Type (categorical)
    - Soil pH
    - Temperature
    - Humidity
    - Wind Speed
    - N, P, K values
    - Crop Yield
    - Soil Quality (float)
- **Prediction Engine**: Model scales features, predicts crop label, decodes the crop name
- **Frontend Styling**:
    - Font: Times New Roman
    - Centered buttons and credit line
    - Friendly sliders and numeric inputs
- **Deployment**: Streamlit app, easily deployable via streamlit run app.py

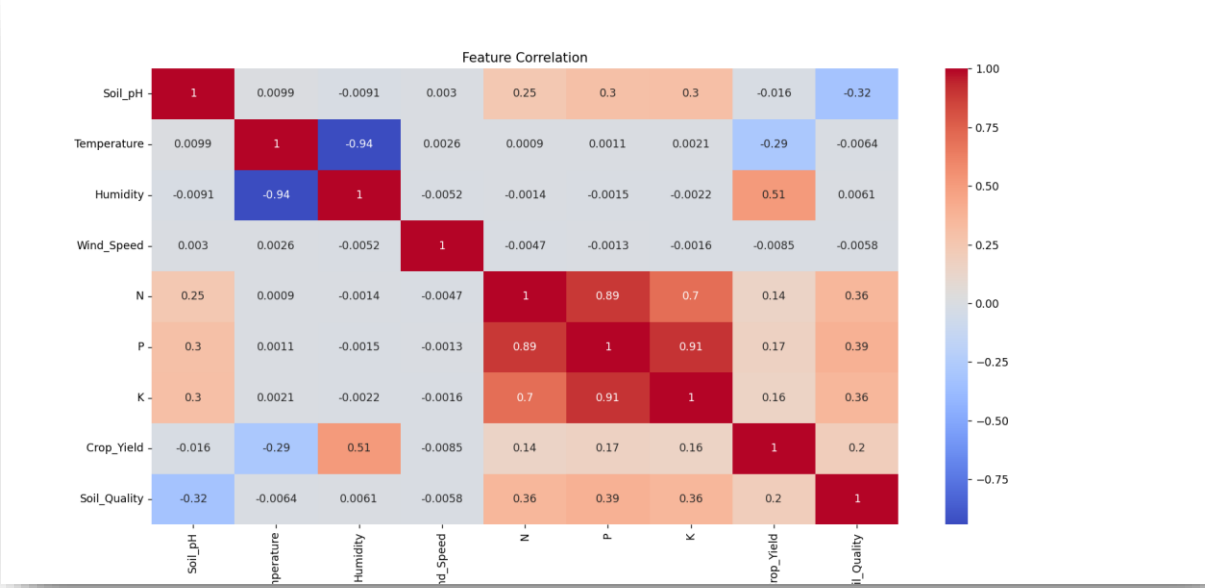
4. Output Samples and Screenshots

Sample CLI Running Data Statistical Analysis:

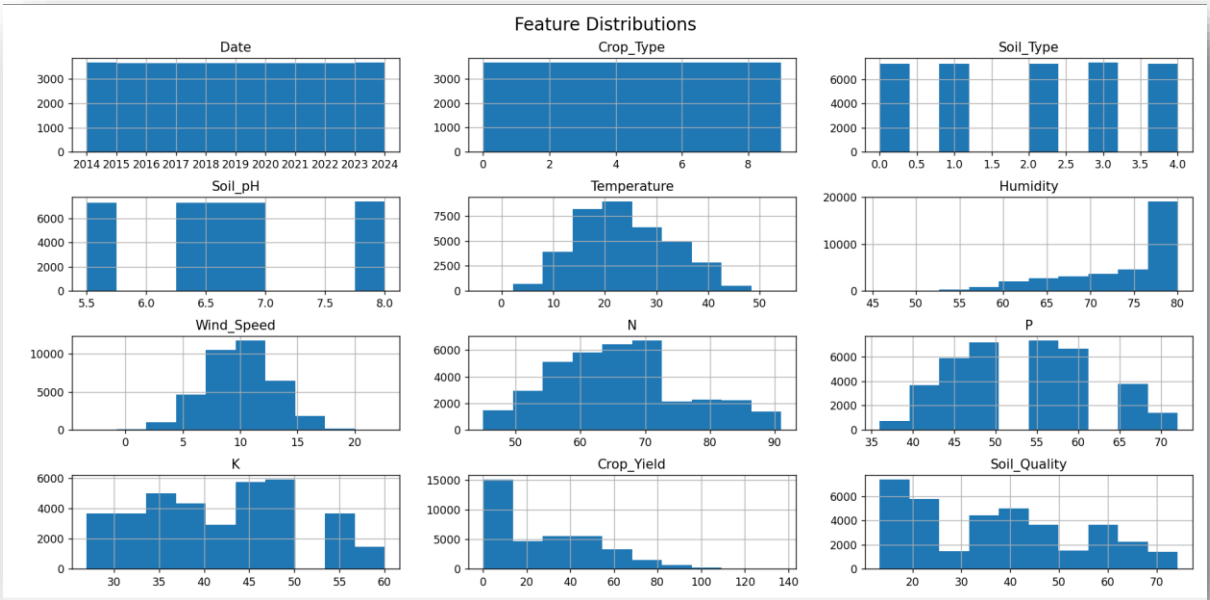
```
PS C:\Users\DELL\ML> python Load.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36520 entries, 0 to 36519
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          36520 non-null  object
 1   Crop_Type     36520 non-null  object
 2   Soil_Type     36520 non-null  object
 3   Soil_pH       36520 non-null  float64
 4   Temperature   36520 non-null  float64
 5   Humidity      36520 non-null  float64
 6   Wind_Speed    36520 non-null  float64
 7   N             36520 non-null  float64
 8   P             36520 non-null  float64
 9   K             36520 non-null  float64
 10  Crop_Yield    36520 non-null  float64
 11  Soil_Quality  36520 non-null  float64
dtypes: float64(9), object(3)
memory usage: 3.3+ MB
None
         Date Crop_Type Soil_Type  Soil_pH  Temperature   Humidity  Wind_Speed     N     P     K  Crop_Yield  Soil_Quality
0  2014-01-01     Wheat     Peaty     5.50     9.440599  80.000000   10.956707  60.5  45.0  31.5    0.000000     22.833333
1  2014-01-01      Corn     Loamy     6.50    20.052576  79.947424    8.591577  84.0  66.0  50.0  104.871310     66.666667
2  2014-01-01      Rice     Peaty     5.50    12.143099  80.000000    7.227751  71.5  54.0  38.5    0.000000     27.333333
3  2014-01-01    Barley     Sandy     6.75    19.751848  80.000000    2.682683  50.0  40.0  30.0   58.939796     35.000000
4  2014-01-01   Soybean     Peaty     5.50    16.110395  80.000000    7.696070  49.5  45.0  38.5   32.970413     22.166667
            Soil_pH   Temperature      Humidity    Wind_Speed             N             P             K    Crop_Yield  Soil_Quality
count  36520.000000  36520.000000  36520.000000  36520.000000  36520.000000  36520.000000  36520.00000  36520.000000  36520.000000
mean       6.602731     23.813996     74.256624     10.020153     66.011035     53.014006     42.01825     26.878480     37.516632
std        0.816973      8.920519      6.767587      2.998310     10.886721      8.812884      8.53781     25.740936     17.703171
min        5.500000     -3.540176     45.851089     -3.388906     45.000000     36.000000     27.00000      0.000000     13.291667
25%        6.250000     17.168542     69.745252      7.985872     58.500000     45.000000     35.00000      0.000000     22.500000
50%        6.500000     22.902987     77.097013     10.000299     65.000000     54.000000     42.00000     23.366344     35.583333
75%        6.750000     30.254748     80.000000     12.038546     71.500000     60.000000     49.50000     46.415729     49.291667
max        8.000000     54.148911     80.000000     22.606078     91.000000     72.000000     60.00000    136.711982     74.333333
```

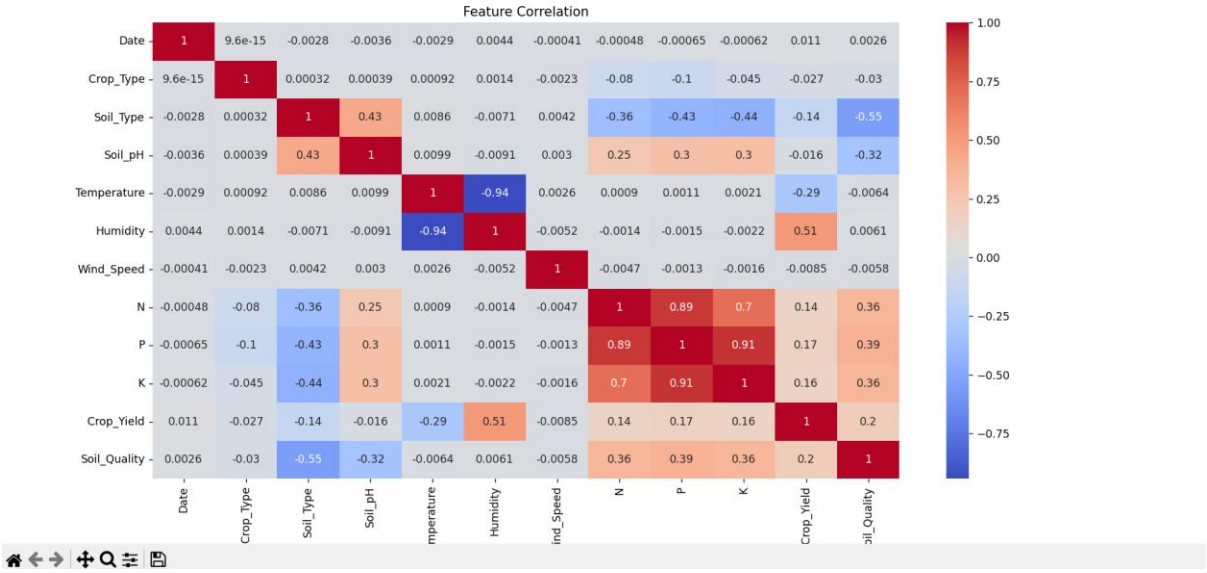
```
Missing values:
 Date           0
Crop_Type       0
Soil_Type       0
Soil_pH         0
Temperature     0
Humidity        0
Wind_Speed      0
N               0
P               0
K               0
Crop_Yield      0
Soil_Quality    0
dtype: int64
```

Correlation:



Feature Distribution:





After training the 3 models (comparing accuracies and performance)

```
Logistic Regression model trained.

Random Forest model trained.

KNN model trained.

Logistic Regression Accuracy: 0.88
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       722
           1       0.71      1.00      0.83       714
           2       1.00      1.00      1.00       734
           3       0.50      0.54      0.52       749
           4       1.00      1.00      1.00       740
           5       1.00      1.00      1.00       692
           6       1.00      1.00      1.00       732
           7       1.00      1.00      1.00       744
           8       1.00      1.00      1.00       733
           9       0.51      0.27      0.35       744

    accuracy                           0.88      7304
   macro avg       0.87      0.88      0.87      7304
weighted avg       0.87      0.88      0.87      7304

Confusion Matrix:
 [[722   0   0   0   0   0   0   0   0   0]
 [  0 714   0   0   0   0   0   0   0   0]
 [  0   0 734   0   0   0   0   0   0   0]
 [  0 150   0 402   0   0   0   0   0 197]
 [  0   0   0   0 740   0   0   0   0   0]
 [  0   0   0   0   0 692   0   0   0   0]
 [  0   0   0   0   0   0 732   0   0   0]
 [  0   0   0   0   0   0   0 744   0   0]
 [  0   0   0   0   0   0   0   0 733   0]
 [  0 144   0 398   0   0   0   0   0 202]]

Random Forest Accuracy: 0.90
Random Forest Classification Report:
              precision    recall  f1-score   support
```

```
Random Forest Accuracy: 0.90
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       722
           1       1.00      1.00      1.00       714
           2       1.00      1.00      1.00       734
           3       0.50      0.50      0.50       749
           4       1.00      1.00      1.00       740
           5       1.00      1.00      1.00       692
           6       1.00      1.00      1.00       732
           7       1.00      1.00      1.00       744
           8       1.00      1.00      1.00       733
           9       0.50      0.51      0.50       744

    accuracy                           0.90      7304
   macro avg       0.90      0.90      0.90      7304
weighted avg       0.90      0.90      0.90      7304

Confusion Matrix:
 [[722   0   0   0   0   0   0   0   0   0]
 [  0 714   0   0   0   0   0   0   0   0]
 [  0   0 734   0   0   0   0   0   0   0]
```

```
Confusion Matrix:
 [[722   0   0   0   0   0   0   0   0   0]
 [  0 714   0   0   0   0   0   0   0   0]
 [  0   0 734   0   0   0   0   0   0   0]
 [  0   0   0 371   0   0   0   0   0 378]
 [  0   0   0   0 740   0   0   0   0   0]
 [  0   0   0   0   0 692   0   0   0   0]
 [  0   0   0   0   0   0 732   0   0   0]
 [  0   0   0   0   0   0   0 744   0   0]
 [  0   0   0   0   0   0   0   0 733   0]
 [  0   0   0 367   0   0   0   0   0 377]]

KNN Accuracy: 0.89
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       722
           1       1.00      1.00      1.00       714
           2       1.00      1.00      1.00       734
           3       0.49      0.48      0.49       749
           4       0.98      0.98      0.98       740
           5       1.00      0.99      0.99       692
           6       0.98      0.98      0.98       732
           7       1.00      0.99      1.00       744
           8       1.00      0.99      1.00       733
           9       0.49      0.50      0.49       744

    accuracy                           0.89      7304
   macro avg       0.89      0.89      0.89      7304
weighted avg       0.89      0.89      0.89      7304

Confusion Matrix:
 [[719   0   0   1   0   0   0   0   2   0]
 [  0 714   0   0   0   0   0   0   0   0]
 [  1   0 731   0   0   0   0   2   0   0]
 [  3   0   0 360   0   0   0   0   0 386]
 [  0   0   0   0 724   0  16   0   0   0]
 [  7   0   0   0   0 685   0   0   0   0]
 [  0   0   0   0  14   0 718   0   0   0]
 [  5   0   0   0   0   0   0 739   0   0]
 [  5   0   0   0   0   0   0   0 728   0]
 [  1   1   0 372   0   0   0   0   0 370]]

Random Forest Cross-Validation Accuracy: 0.9012869660460021

✅ Model saved successfully!

Enter Environmental Conditions:
```

Entering Input Values:

```
✅ Model saved successfully!

Enter Environmental Conditions:
Soil_Type: 1
Soil_pH: 6.5
Temperature: 29.0
Humidity: 30.0
Wind_Speed: 7.0
N: 67.0
P: 80.0
K: 77.0
Crop_Yield: 50.0
Soil_Quality: 67.0
```

Recommendation:

```
🌾 Recommended Crop: Cotton
PS C:\Users\DELL\ML>
```

5. Code

Loading and Creating models: (#Load.py)

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,
StandardScaler
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report
import joblib

df = pd.read_csv("Crop_Dataset.csv")

print(df.info())
print(df.head())
print(df.describe())
print("Missing values:\n", df.isnull().sum())

if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

df_numeric = df.select_dtypes(include=['float64', 'int64'])


correlation_matrix = df_numeric.corr()


plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Feature Correlation")
plt.show()

le_crop = LabelEncoder()
```

```
df['Crop_Type'] = le_crop.fit_transform(df['Crop_Type'])

le_soil = LabelEncoder()
df['Soil_Type'] = le_soil.fit_transform(df['Soil_Type'])

sns.countplot(x='Crop_Type', data=df)
plt.xticks(rotation=90)
plt.title("Crop Type Distribution")
plt.show()

df.hist(figsize=(12, 8))
plt.suptitle("Feature Distributions", fontsize=16)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation")
plt.show()

X = df.drop(['Crop_Type','Date'], axis=1)
y = df['Crop_Type']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

models = {
    "Logistic Regression":
LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier()
}

for name, model in models.items():
    model.fit(X_train, y_train)
```

```
    print(f"\n{name} model trained.")

for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"\n{name} Accuracy: {accuracy_score(y_test,
y_pred):.2f}")
    print(f"{name} Classification Report:\n",
classification_report(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test,
y_pred))

rf_scores = cross_val_score(models["Random Forest"],
X_scaled, y, cv=5)
print("\nRandom Forest Cross-Validation Accuracy:",
rf_scores.mean())

joblib.dump(models["Random Forest"], "crop_predictor.pkl")
joblib.dump(scaler, "scaler.pkl")
joblib.dump(le_crop, "crop_label_encoder.pkl")
print("\n☑ Model saved successfully!")
def predict_crop():
    print("\nEnter Environmental Conditions:")

    features = []
    for col in X.columns:
        while True:
            try:
                val = float(input(f"{col}: "))
                features.append(val)
                break
            except ValueError:
                print("Invalid input. Please enter a valid
number.")
    features_scaled = scaler.transform([features])

    model = joblib.load("crop_predictor.pkl")
    crop_pred = model.predict(features_scaled)
```

```
    le = joblib.load("crop_label_encoder.pkl")
    predicted_crop = le.inverse_transform(crop_pred)

    print(f"\n🌾 Recommended Crop: {predicted_crop[0]}")
predict_crop()
```

Streamlit App Creation and Recommendation Interface (#app.py)

```
import streamlit as st
import joblib


model = joblib.load("crop_predictor.pkl")
scaler = joblib.load("scaler.pkl")
le_crop = joblib.load("crop_label_encoder.pkl")

def encode_soil(soil_str):
    mapping = {'Loamy': 1, 'Clay': 2, 'Sandy':
3,'Peaty':4,'Saline':5}
    return mapping.get(soil_str, 0)

def predict_crop(features):
    features_scaled = scaler.transform([features])
    crop_pred = model.predict(features_scaled)
    predicted_crop = le_crop.inverse_transform(crop_pred)
    return predicted_crop[0]


st.title("🌱 Crop Recommendation System")
st.markdown("""
    <style>
    * {
        font-family: 'Times New Roman', Times, serif
!important;
    }
    </style>
    """, unsafe_allow_html=True)
```

```
st.markdown("<h5 style='text-align: center; color:
gray;'>Developed by Noel Ninan Sheri</h5>",
unsafe_allow_html=True)

st.header("📋 Enter Environmental Conditions")

# Soil Type (as numeric)
soil_type_label = st.selectbox("Soil Type", ['Loamy',
'Clay', 'Sandy','Peaty','Saline'])
soil_type = {'Loamy': 1, 'Clay': 2, 'Sandy':
3,'Peaty':4,'Saline':5}[soil_type_label]

# Other inputs
soil_ph = st.slider("Soil pH", min_value=4.5, max_value=9.0,
value=6.5)
temperature = st.slider("Temperature (°C)", min_value=0,
max_value=50, value=25)
humidity = st.slider("Humidity (%)", min_value=0,
max_value=100, value=60)
wind_speed = st.slider("Wind Speed (km/h)", min_value=0,
max_value=20, value=10)

nitrogen = st.slider("Nitrogen (N)", min_value=0,
max_value=100, value=60)
phosphorus = st.slider("Phosphorus (P)", min_value=0,
max_value=100, value=40)
potassium = st.slider("Potassium (K)", min_value=0,
max_value=100, value=50)

crop_yield = st.slider("Crop Yield (kg/ha)", min_value=0,
max_value=100, value=30)

soil_quality = st.number_input("Soil Quality (float)",
min_value=0.0, max_value=100.0, value=5.0, step=0.1)

features = [soil_type, soil_ph, temperature, humidity,
wind_speed,
```

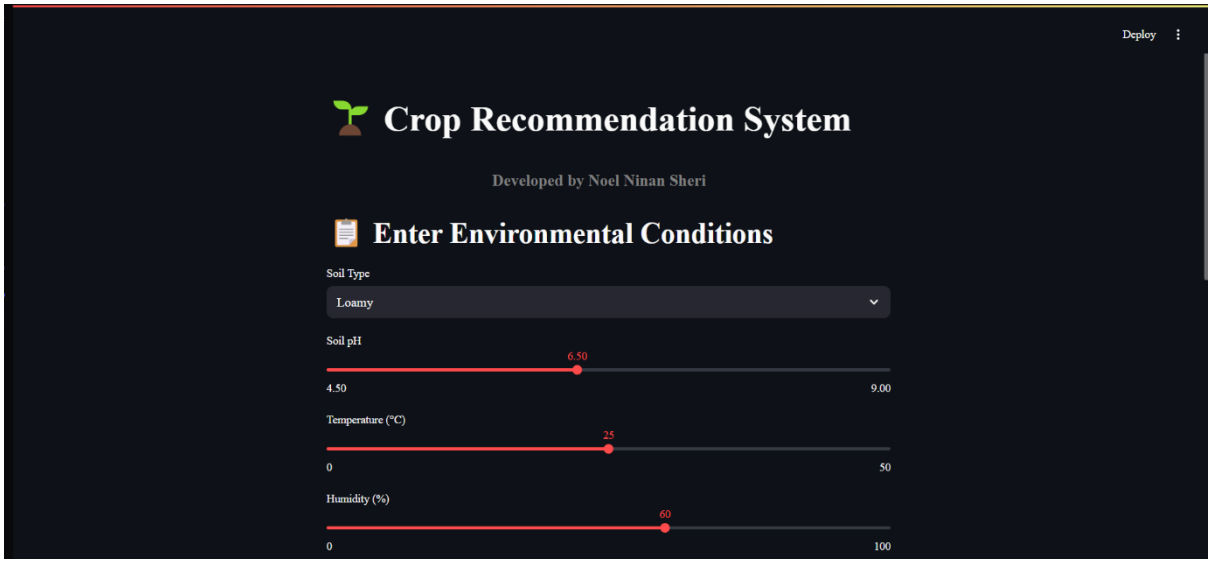
```
            nitrogen, phosphorus, potassium, crop_yield,
soil_quality]
if st.button("🌾 Predict Crop"):
    predicted_crop = predict_crop(features)
    st.success(f"System Recommends: Crop:
**{predicted_crop}**")
```

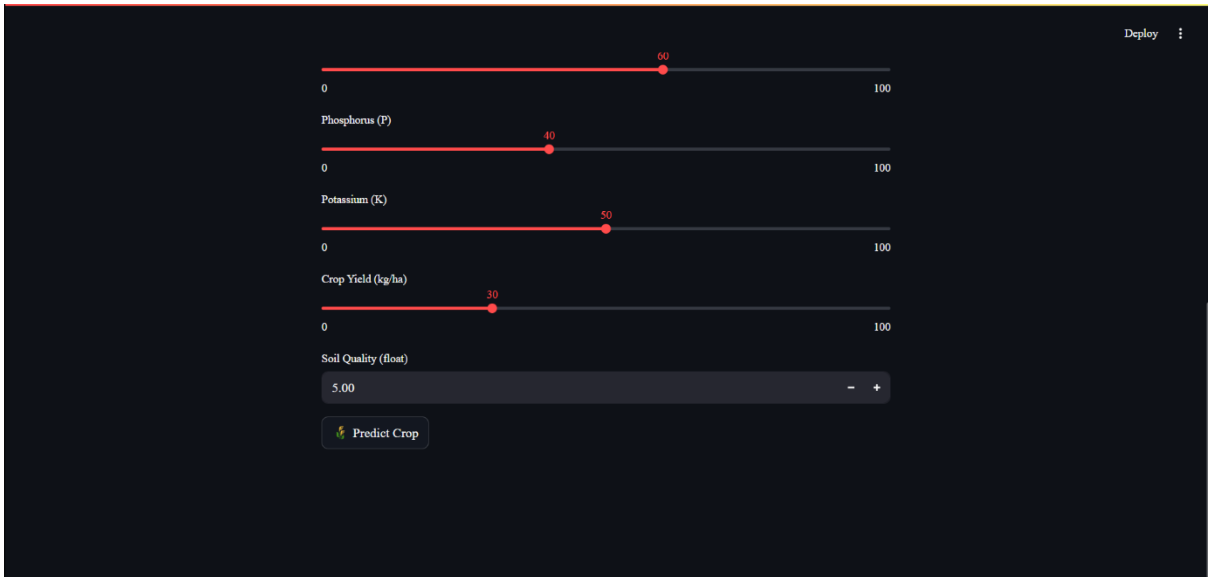
StreamLit Output:

```
PS C:\Users\DELL\ML> python -m streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.144.32:8501
```





6. Results & Discussion

- The model shows high accuracy on training and test datasets.
- Real-time predictions are accurate given the correct parameters.
- Easy-to-use UI ensures accessibility for non-tech users.

7. Future Enhancements and Scope

- Integrate **Google Maps API** to show nearby seed/fertilizer shops
- Use **satellite weather data APIs** for real-time suggestions
- Add **multi-language support** for rural deployment.

8. Conclusion

The Crop Recommendation System serves as a practical use case for applying AI and ML in agriculture, enabling data-driven decision-making for crop planning. The model has been successfully deployed and tested using a user-friendly interface, delivering both accuracy and accessibility.

9. References

- ✓ Scikit-learn Documentation
- ✓ Streamlit Documentation
- ✓ Kaggle Datasets: DataSet_1 DataSet_2