Google Colab Code

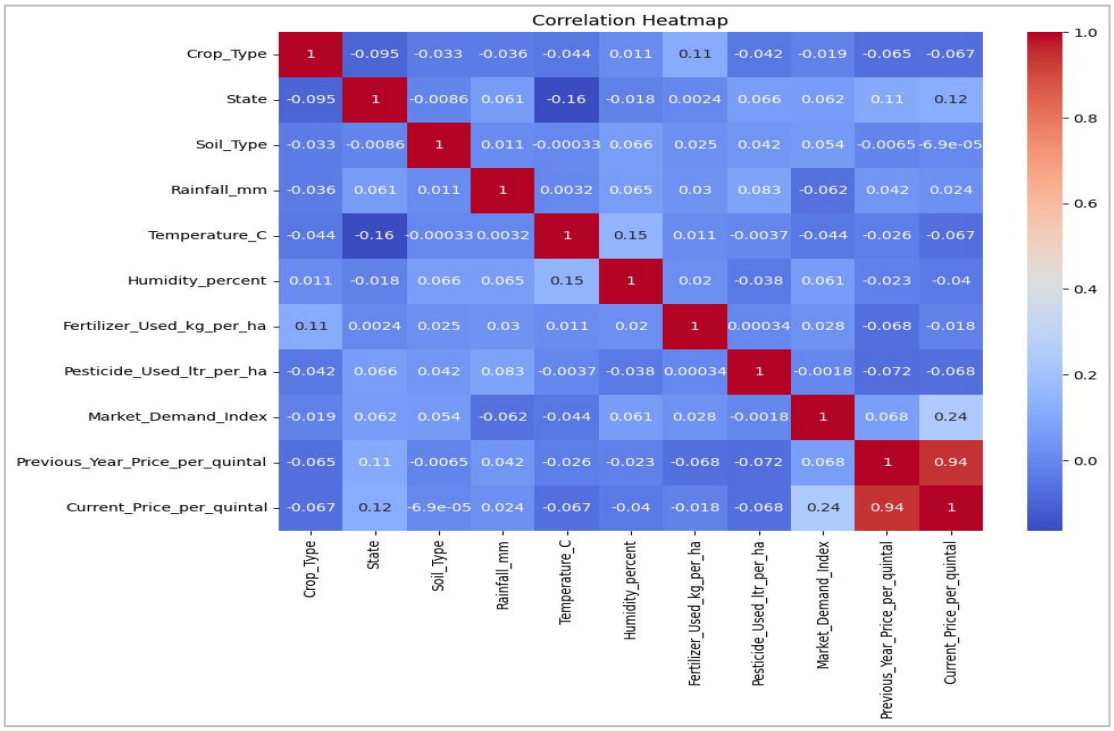
1. code for finding the accuracy of Decision Tree, Random Forest, and Linear Regression, and then plot the heatmap.

```
# LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,
        r2_score, mean_absolute_error, accuracy_score, classification_report, roc_curve, auc

# LOAD DATASET
from google.colab import files
uploaded = files.upload()
df = pd.read_csv("crop_price_prediction_dataset.csv")
print("\nDataset Loaded:")
print(df.head())

# ENCODE CATEGORICAL DATA
le = LabelEncoder()
df['Crop_Type'] = le.fit_transform(df['Crop_Type'])
df['State'] = le.fit_transform(df['State'])
df['Soil_Type'] = le.fit_transform(df['Soil_Type'])

# FEATURES AND TARGET
X = df.drop(columns=['Current_Price_per_quintal'])
y = df['Current_Price_per_quintal']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# STEP 6: SCALING
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# MODELS
models = {
  "Linear Regression": LinearRegression(), # Use LinearRegression for regression
  "Decision Tree": DecisionTreeRegressor(),
  "Random Forest": RandomForestRegressor(n_estimators=100),
}

# TRAIN AND EVALUATE
model_scores = {}
```

----------*****----------

```
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    r2 = r2_score(y_test, y_pred)
    model_scores[name] = r2
    print(f"\n{name} Results:")
    print(f"R² Score (Accuracy): {r2:.4f}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
    print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}")
    print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred)):.2f}")

# CORRELATION HEATMAP
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

```
Linear Regression Results:
R² Score (Accuracy): 0.9315
MAE: 71.52
MSE: 7733.38
RMSE: 87.94

Decision Tree Results:
R² Score (Accuracy): 0.8377
MAE: 104.31
MSE: 18324.36
RMSE: 135.37

Random Forest Results:
R² Score (Accuracy): 0.9180
MAE: 75.76
MSE: 9254.46
RMSE: 96.20
```



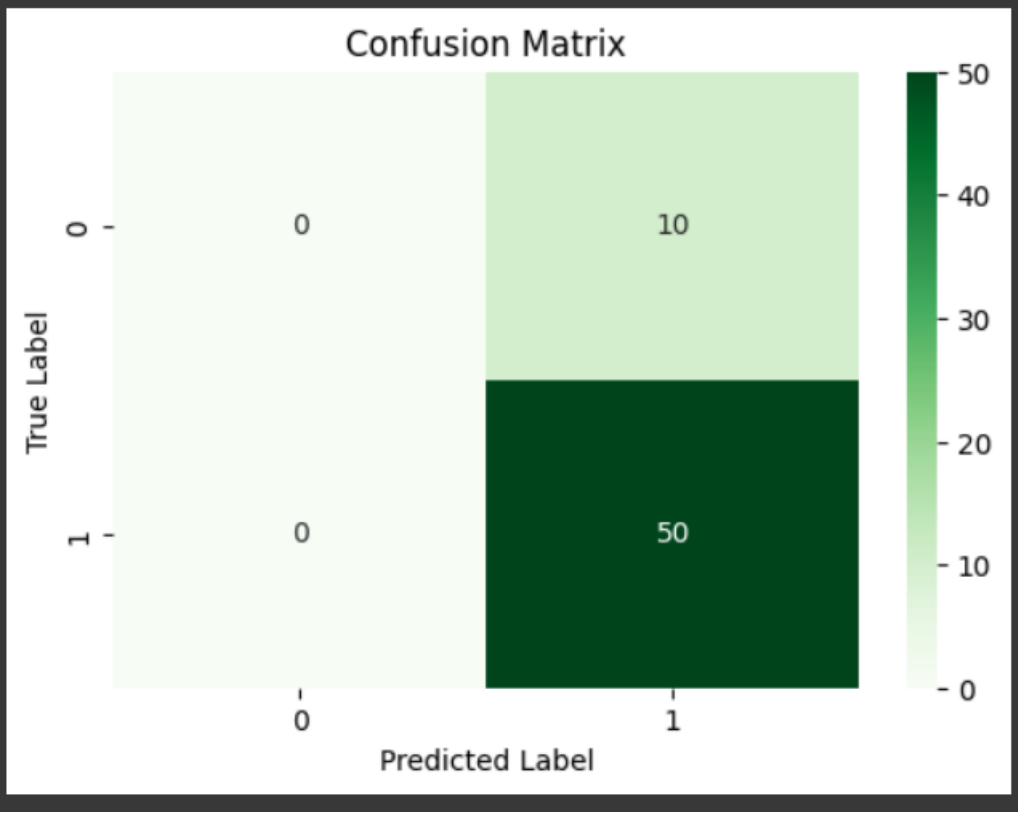
2. Plot confusion matrix with linear regression?

```
# LIBARARIES
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('crop_price_prediction_dataset.csv')
print("Dataset Preview:")
print(df.head())

# Data Cleaning
print("\nMissing values:")
print(df.isnull().sum())
df.dropna(inplace=True)

# Encode categorical variables
from sklearn.preprocessing import LabelEncoder
for column in df.select_dtypes(include='object').columns:
  le = LabelEncoder()
  df[column] = le.fit_transform(df[column])

# Features and target
df['Price_Change'] = (df['Current_Price_per_quintal'] >
df['Previous_Year_Price_per_quintal']).astype(int)
y = df['Price_Change']

# Train-Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model
model = LogisticRegression(max_iter=1000)
y_pred = model.predict(X_test)

# Evaluation
print(f"\nAccuracy: {accuracy_score(y_test, y_pred):.2f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix Visualization
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
```

----------*****----------

```
sns.heatmap(cm, annot=True, fmt="d", cmap="Greens")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.83

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        10
           1       0.83      1.00      0.91        50

    accuracy                           0.83        60
   macro avg       0.42      0.50      0.45        60
weighted avg       0.69      0.83      0.76        60
```



---*****---

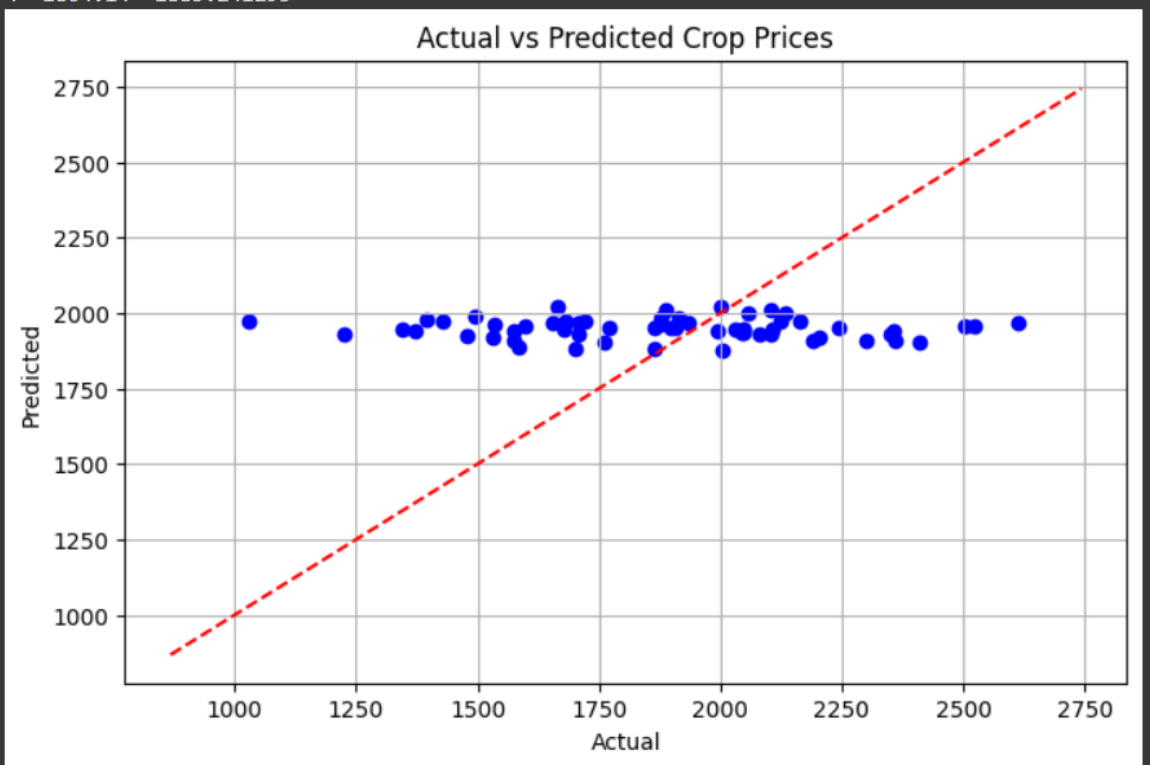
3. Linear regression

```
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('crop_price_prediction_dataset.csv')
print("\nDataset Preview:")
print(df.head())

# clean the data
print("\nMissing Values:")
print(df.isnull().sum())

# Drop rows with missing values (you can also use fillna if needed)
df = df.dropna()

# Display column names
print("\nColumns in the dataset:")
print(df.columns)

# Define features and target
features = ['Rainfall_mm', 'Temperature_C', 'Fertilizer_Used_kg_per_ha']
target = 'Current_Price_per_quintal'

X = df[features]
y = df[target]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# Evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"\nMean Squared Error (MSE): {mse:.2f}")
print(f"R-squared Score (R2): {r2:.2f}")

# Compare actual vs predicted values
results = pd.DataFrame({'Actual': y_test.values, 'Predicted': y_pred})
print("\nComparison of Actual vs Predicted:")
print(results.head())

# Visualization
```

----------*****----------

```
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted Crop Prices')
plt.grid(True)
plt.show()
```

```
Mean Squared Error (MSE): 118923.83
R-squared Score (R2): -0.05

Comparison of Actual vs Predicted:
    Actual    Predicted
0  1914.68  1981.839391
1  2411.02  1904.658048
2  1906.10  1949.388292
3  2298.86  1909.886170
4  1864.14  1883.241295
```



---*****---

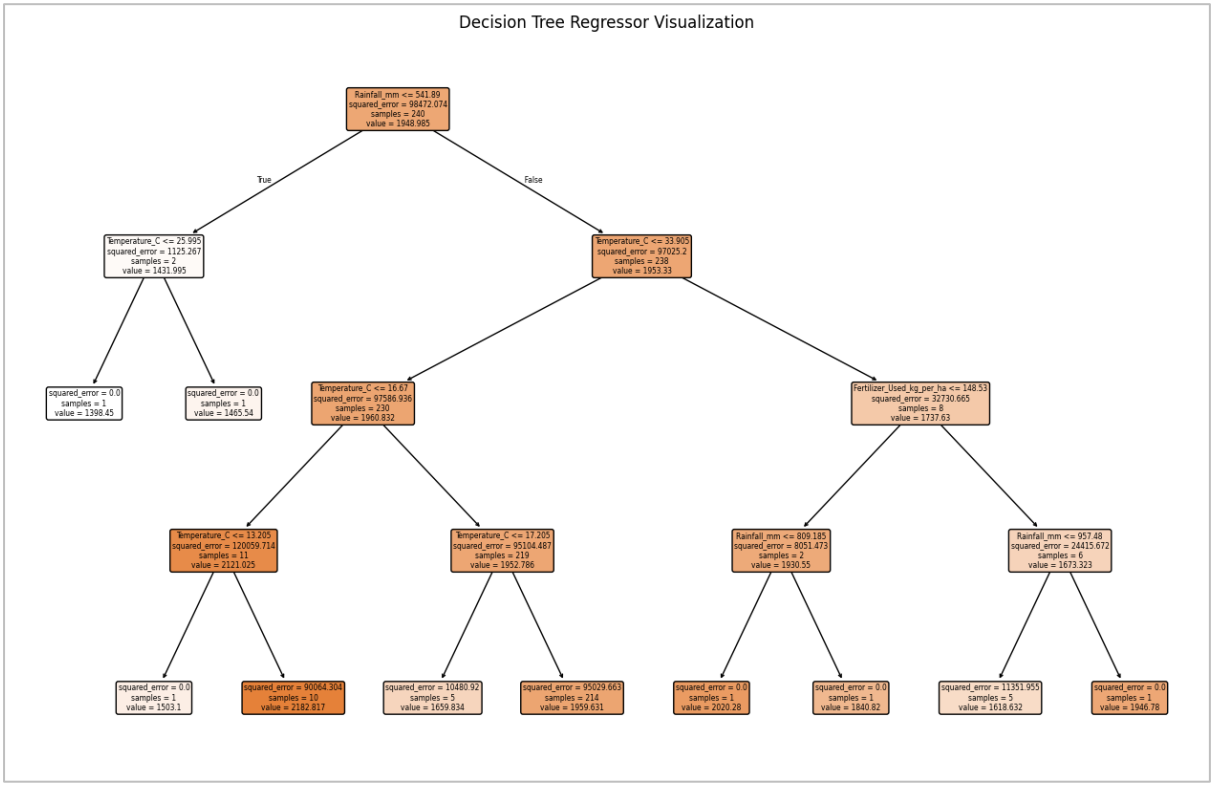
4. Decision tree

```
# question.....

# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.model_selection import train_test_split

# Load dataset
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('crop_price_prediction_dataset.csv')
print("\nDataset preview:")
print(df.head())

# Check for nulls
print("\nMissing values in the dataset:")
print(df.isnull().sum())
df = df.dropna()

# feature/target selection
print("\nColumns in the dataset:")
print(df.columns)

# Select features and target
features = ['Rainfall_mm', 'Temperature_C', 'Fertilizer_Used_kg_per_ha']
target = 'Current_Price_per_quintal'

X = df[features]
y = df[target]

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

dt_model = DecisionTreeRegressor(max_depth=4)
dt_model.fit(X_train, y_train)

y_pred = dt_model.predict(X_test)

# Display predicted vs actual
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print("\nPrediction Results:")
print(results.head())

# Visualize the decision tree
plt.figure(figsize=(16, 10))
```

----------*****----------

```
plot_tree(dt_model, feature_names=features, filled=True, rounded=True)
plt.title("Decision Tree Regressor Visualization")
plt.show()
```

```
Columns in the dataset:
Index(['Crop_Type', 'State', 'Soil_Type', 'Rainfall_mm', 'Temperature_C',
       'Humidity_percent', 'Fertilizer_Used_kg_per_ha',
       'Pesticide_Used_ltr_per_ha', 'Market_Demand_Index',
       'Previous_Year_Price_per_quintal', 'Current_Price_per_quintal'],
      dtype='object')

Prediction Results:
      Actual    Predicted
203  1914.68  1959.630654
266  2411.02  1959.630654
152  1906.10  1959.630654
9    2298.86  1959.630654
233  1864.14  1959.630654
```



---*****---

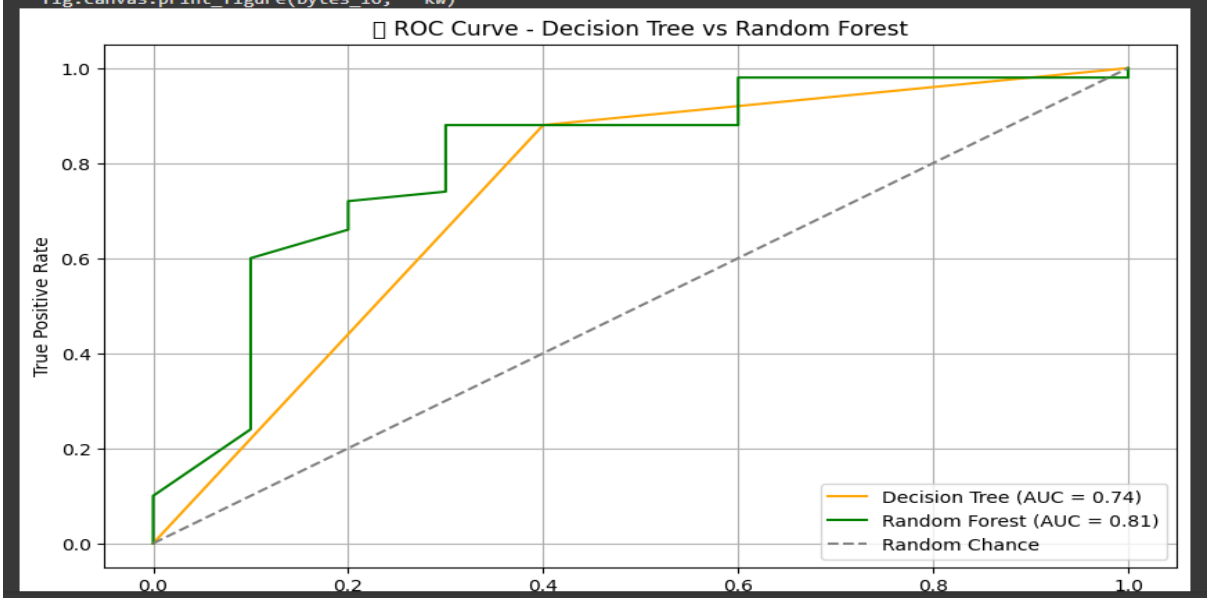
5. Code for ROC and AUC curves for the Random Forest and Decision Tree models, and perform feature selection using the Random Forest model.

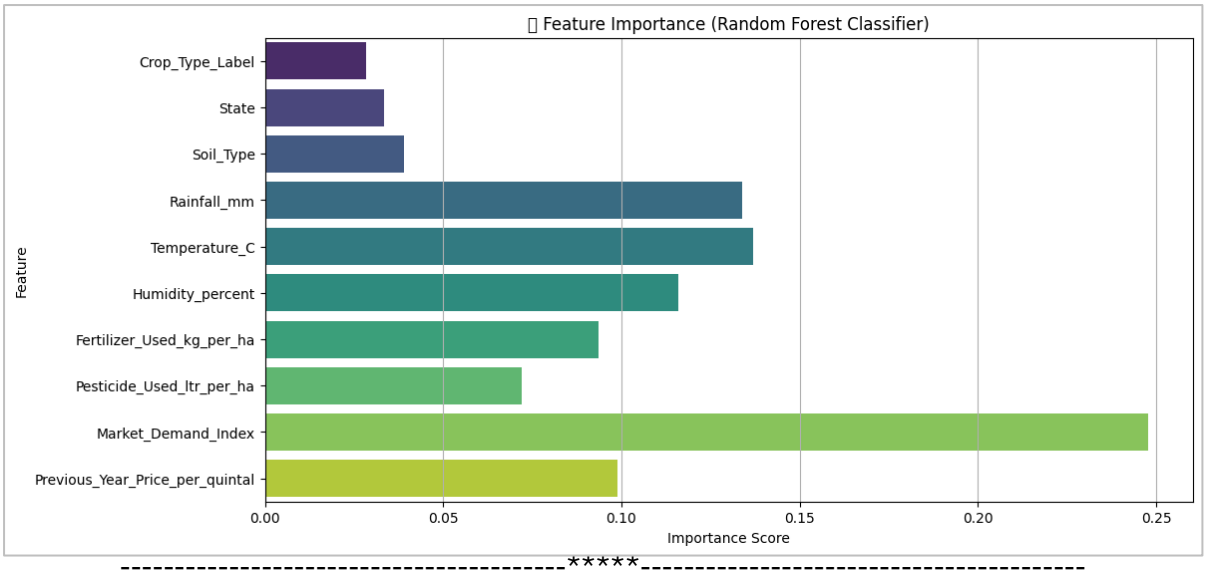
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, roc_curve, auc
import matplotlib.pyplot as plt

# Split the data again
X_train_dt, X_test_dt, y_train_dt, y_test_dt = train_test_split(X_scaled, y_class, test_size=0.2, random_state=42)

# Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train_dt, y_train_dt)
y_pred_dt = dt_model.predict(X_test_dt)
y_prob_dt = dt_model.predict_proba(X_test_dt)[:, 1]

# Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_dt, y_train_dt)
y_pred_rf = rf_model.predict(X_test_dt)
y_prob_rf = rf_model.predict_proba(X_test_dt)[:, 1]

# Decision Tree Metrics
print(f" 🌳 Decision Tree Accuracy: {accuracy_score(y_test_dt, y_pred_dt):.2f}")
print("\n 📋 Classification Report (Decision Tree):\n", classification_report(y_test_dt, y_pred_dt))
fpr_dt, tpr_dt, _ = roc_curve(y_test_dt, y_prob_dt)
auc_dt = auc(fpr_dt, tpr_dt)
print(f" 📈 AUC Score (Decision Tree): {auc_dt:.2f}")

# Random Forest Metrics
print(f"\n 🎄 Random Forest Accuracy: {accuracy_score(y_test_dt, y_pred_rf):.2f}")
print("\n 📋 Classification Report (Random Forest):\n", classification_report(y_test_dt, y_pred_rf))
fpr_rf, tpr_rf, _ = roc_curve(y_test_dt, y_prob_rf)
auc_rf = auc(fpr_rf, tpr_rf)
print(f" 📈 AUC Score (Random Forest): {auc_rf:.2f}")

# Plot ROC Curves for both
plt.figure(figsize=(10, 6))
plt.plot(fpr_dt, tpr_dt, label=f"Decision Tree (AUC = {auc_dt:.2f})", color='orange')
plt.plot(fpr_rf, tpr_rf, label=f"Random Forest (AUC = {auc_rf:.2f})", color='green')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Chance')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title(" 🌾 ROC Curve - Decision Tree vs Random Forest")
plt.legend(loc='lower right')
```

----------*****----------

```
plt.grid(True)
plt.show()

# Feature Importance from Random Forest Classifier
plt.figure(figsize=(12, 6))
sns.barplot(x=importances, y=feature_names, palette='viridis')
plt.title("🎯 Feature Importance (Random Forest Classifier)")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.grid(axis='x')
plt.show()
```

```
🌲 Random Forest Accuracy: 0.82

📄 Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        10
           1       0.83      0.98      0.90        50

    accuracy                           0.82        60
   macro avg       0.42      0.49      0.45        60
weighted avg       0.69      0.82      0.75        60

📈 AUC Score (Random Forest): 0.81
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 127806 (\N{EAR OF RI
  fig.canvas.print_figure(bytes_io, **kw)
```





----------*****----------

6. Final prediction code

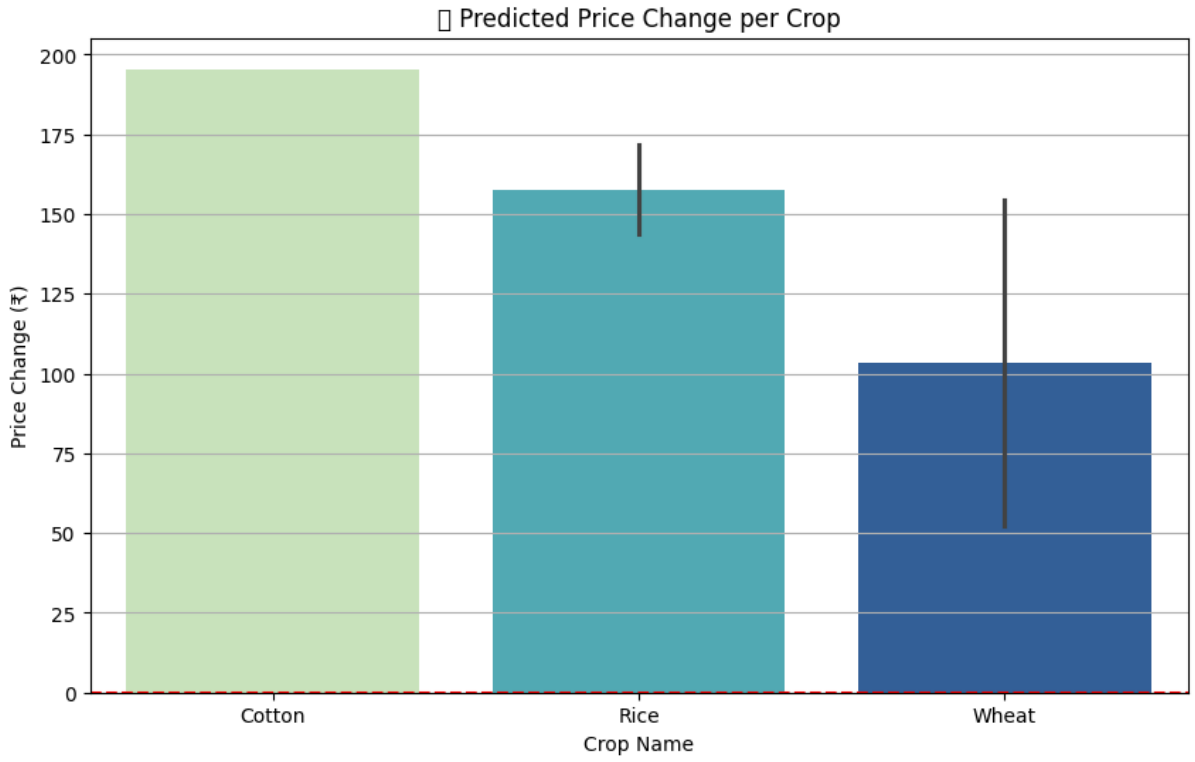
```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, classification_report, r2_score
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset
from google.colab import files
uploaded = files.upload()
df = pd.read_csv("crop_price_prediction_dataset.csv")
print("Dataset preview:\n", df.head())

# Encode categorical variables
le = LabelEncoder()
df['Crop_Type_Label'] = le.fit_transform(df['Crop_Type'])
df['State'] = LabelEncoder().fit_transform(df['State'])
df['Soil_Type'] = LabelEncoder().fit_transform(df['Soil_Type'])

# Create classification label and regression target
df['Price_Increase'] = (df['Current_Price_per_quintal'] > df['Previous_Year_Price_per_quintal']).astype(int)
df['Price_Diff'] = (df['Current_Price_per_quintal'] - df['Previous_Year_Price_per_quintal']).round(2)

# Features for classification and regression
feature_cols = ['Crop_Type_Label', 'State', 'Soil_Type', 'Rainfall_mm', 'Temperature_C',
        'Humidity_percent', 'Fertilizer_Used_kg_per_ha', 'Pesticide_Used_ltr_per_ha',
        'Market_Demand_Index', 'Previous_Year_Price_per_quintal']

X = df[feature_cols]
y_class = df['Price_Increase']
y_reg = df['Price_Diff']

# Step 7: Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 8: Split data
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_scaled, y_class, test_size=0.2, random_state=42)
X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X_scaled, y_reg, test_size=0.2, random_state=42)

clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_c, y_train_c)
```

----------*****----------

```
reg = RandomForestRegressor(random_state=42)
reg.fit(X_train_r, y_train_r)

# Evaluate classification regression
y_pred_c = clf.predict(X_test_c)
print(f"\n 📊 Classification Accuracy: {accuracy_score(y_test_c, y_pred_c):.2f}")
print("\n 📋 Classification Report:\n", classification_report(y_test_c, y_pred_c))
y_pred_r = reg.predict(X_test_r)
print(f" 📈 Regression R² Score: {r2_score(y_test_r, y_pred_r):.2f}")

# Make prediction on new data
print("\n 🔮 --- Future Crop Predictions ---")
future_data = df.sample(5).copy()  # simulate unseen crops
future_input = scaler.transform(future_data[feature_cols])
pred_class = clf.predict(future_input)
pred_diff = reg.predict(future_input)

for i in range(len(future_data)):
    crop_type_label = int(future_data.iloc[i]['Crop_Type_Label'])
    crop_name = le.inverse_transform([crop_type_label])[0]

    previous_price = future_data.iloc[i]['Previous_Year_Price_per_quintal']
    increase = "↑ Increase" if pred_class[i] == 1 else "→ No Increase"
    diff = f"{pred_diff[i]:.2f} ₹"
    new_price = previous_price + pred_diff[i]

    print(f"Crop: {crop_name}")
    print(f" - Previous Price: ₹{previous_price}")
    print(f" - Prediction: {increase}")
    print(f" - Predicted Increase: ₹{diff}")
    print(f" - Expected New Price: ₹{new_price:.2f}")
    print("-" * 40)

#  Predicted Price Increase (Bar Chart)
plt.figure(figsize=(10, 6))
sns.barplot(x=crop_names, y=price_increase_values, palette='YlGnBu')
plt.title(" 🌾  Predicted Price Change per Crop")
plt.ylabel("Price Change (₹)")
plt.xlabel("Crop Name")
plt.axhline(0, color='red', linestyle='--')
plt.grid(axis='y')
plt.show()
```

----------*****----------

```
📊 Classification Accuracy: 0.82

📋 Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        10
           1       0.83      0.98      0.90        50

    accuracy                           0.82        60
   macro avg       0.42      0.49      0.45        60
weighted avg       0.69      0.82      0.75        60

📈 Regression R² Score: 0.34

🔮 --- Future Crop Predictions ---
Crop: Cotton
 - Previous Price: ₹1561.15
 - Prediction: ↑ Increase
 - Predicted Increase: ₹195.43 ₹
 - Expected New Price: ₹1756.58
----------------------------------------
Crop: Rice
 - Previous Price: ₹1601.16
 - Prediction: ↑ Increase
 - Predicted Increase: ₹143.60 ₹
 - Expected New Price: ₹1744.76
----------------------------------------
Crop: Wheat
 - Previous Price: ₹2002.98
 - Prediction: ↑ Increase
 - Predicted Increase: ₹52.41 ₹
 - Expected New Price: ₹2055.39
----------------------------------------
```



----------*****----------