! pip install requests pandas matplotlib scikit-learn  
import requests

import pandas as pd

df = pd.read\_csv("/content/drive/MyDrive/commodity\_prices.csv", dtype=str)

df["Min Price"] = pd.to\_numeric(df["Min Price"], errors="coerce")

df["Max Price"] = pd.to\_numeric(df["Max Price"], errors="coerce")

df["Modal Price"] = pd.to\_numeric(df["Modal Price"], errors="coerce")

*# Convert "Arrival Date" to datetime format*

df["Arrival Date"] = pd.to\_datetime(df["Arrival Date"], errors="coerce")

*# Check data types again*

print(df.dtypes)

df

*# Divide the selected columns by 100*

df[columns\_to\_modify] = df[columns\_to\_modify] / 100

target = "Modal Price" *# Predicting the "Modal Price"*

features = ["State", "District", "Market", "Commodity", "Variety", "Grade", "Arrival Date", "Min Price", "Max Price"]

df = df.dropna(subset=[target])

df

from sklearn.preprocessing import LabelEncoder

label\_encoders = {} *# Store encoders for future use*

for col in ["State", "District", "Market", "Commodity", "Variety", "Grade"]:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col].astype(str)) *# Convert categorical to numerical*

label\_encoders[col] = le

df["Arrival Date"] = (df["Arrival Date"] - df["Arrival Date"].min()).dt.days

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

X = df[features]

y = df[target]

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

from sklearn.ensemble import RandomForestRegressor

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

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

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

from sklearn.metrics import mean\_absolute\_error, r2\_score

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae:.2f}")

print(f"R² Score: {r2:.2f}")

import pandas as pd

from sklearn.preprocessing import OrdinalEncoder

import numpy as np

*# Assuming the encoder is already fitted during training*

encoder = OrdinalEncoder(handle\_unknown='use\_encoded\_value', unknown\_value=-1)

*# Define the prediction function*

def predict\_price(state, district, market, commodity, variety, grade, arrival\_date, min\_price, max\_price):

*# Encode categorical values using the encoder*

encoded\_features = []

for feature, value in zip(["State", "District", "Market", "Commodity", "Variety", "Grade"],

[state, district, market, commodity, variety, grade]):

if value in label\_encoders[feature].classes\_:

encoded\_features.append(label\_encoders[feature].transform([value])[0])

else:

encoded\_features.append(-1) *# Fallback for unseen values*

*# Convert Arrival Date to number of days since the minimum arrival date*

encoded\_features.append((pd.to\_datetime(arrival\_date) - pd.to\_datetime(df["Arrival Date"].min())).days)

encoded\_features.extend([min\_price, max\_price])

*# Convert the feature list to a DataFrame with the same column names as during training*

feature\_columns = ["State", "District", "Market", "Commodity", "Variety", "Grade", "Arrival Date", "Min Price", "Max Price"]

X\_new = pd.DataFrame([encoded\_features], columns=feature\_columns)

*# Make the prediction*

predicted\_price = model.predict(X\_new)[0]

*# Return predicted price only*

return {"Predicted Price": round(predicted\_price, 2)}

*# Example Prediction*

predicted = predict\_price("Tamil Nadu", "Ariyalur", "Ariyalur", "Banana", "Besrai", "Local", "2025-01-03", 54.0, 64.0)

print(predicted)