import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read\_csv('yield\_df.csv')

df.head()

df.drop('Unnamed: 0',axis=1,inplace=True)

df.shape

df.info()

df.isnull().sum()

df.duplicated().sum()

df.drop\_duplicates(inplace=True)

df.duplicated().sum()

def isStr(obj):

    try:

        float(obj)

        return False

    except:

        return True

to\_drop = df[df['average\_rain\_fall\_mm\_per\_year'].apply(isStr)].index

df = df.drop(to\_drop)

df

df['average\_rain\_fall\_mm\_per\_year'] = df['average\_rain\_fall\_mm\_per\_year'].astype(np.float64)

len(df['Area'].unique())

plt.figure(figsize=(15,20))

sns.countplot(y=df['Area'])

plt.show()

(df['Area'].value\_counts() < 500).sum()

country = df['Area'].unique()

yield\_per\_country = []

for state in country:

    yield\_per\_country.append(df[df['Area']==state]['hg/ha\_yield'].sum())

df['hg/ha\_yield'].sum()

yield\_per\_country

plt.figure(figsize=(15, 20))

sns.barplot(y=country, x=yield\_per\_country)

sns.countplot(y=df['Item'])

crops = df['Item'].unique()

yield\_per\_crop = []

for crop in crops:

    yield\_per\_crop.append(df[df['Item']==crop]['hg/ha\_yield'].sum())

sns.barplot(y=crops,x=yield\_per\_crop)

col = ['Year', 'average\_rain\_fall\_mm\_per\_year','pesticides\_tonnes', 'avg\_temp', 'Area', 'Item', 'hg/ha\_yield']

df = df[col]

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

df.head(3)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8, random\_state=0, shuffle=True)

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler

ohe = OneHotEncoder(drop='first')

scale = StandardScaler()

preprocesser = ColumnTransformer(

        transformers = [

            ('StandardScale', scale, [0, 1, 2, 3]),

            ('OHE', ohe, [4, 5]),

        ],

        remainder='passthrough'

)

X\_train\_dummy = preprocesser.fit\_transform(X\_train)

X\_test\_dummy = preprocesser.transform(X\_test)

preprocesser.get\_feature\_names\_out(col[:-1])

#linear regression

from sklearn.linear\_model import LinearRegression,Lasso,Ridge

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

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

models = {

    'lr':LinearRegression(),

    'lss':Lasso(),

    'Rid':Ridge(),

    'Dtr':DecisionTreeRegressor()

}

for name, md in models.items():

    md.fit(X\_train\_dummy,y\_train)

    y\_pred = md.predict(X\_test\_dummy)

    print(f"{name} : mae : {mean\_absolute\_error(y\_test,y\_pred)} score : {r2\_score(y\_test,y\_pred)}")

dtr = DecisionTreeRegressor()

dtr.fit(X\_train\_dummy,y\_train)

dtr.predict(X\_test\_dummy)

def prediction(Year, average\_rain\_fall\_mm\_per\_year, pesticides\_tonnes, avg\_temp, Area, Item):

    # Create an array of the input features

    features = np.array([[Year, average\_rain\_fall\_mm\_per\_year, pesticides\_tonnes, avg\_temp, Area, Item]], dtype=object)

    # Transform the features using the preprocessor

    transformed\_features = preprocesser.transform(features)

    # Make the prediction

    predicted\_yield = dtr.predict(transformed\_features).reshape(1, -1)

    return predicted\_yield[0]

Year = 1990

average\_rain\_fall\_mm\_per\_year =1485.0

pesticides\_tonnes = 121.00

avg\_temp = 16.37

Area = 'Albania'

Item = 'Maize'

result = prediction(Year, average\_rain\_fall\_mm\_per\_year, pesticides\_tonnes, avg\_temp, Area, Item)

result

entry1 = (1990, 1485.0, 121.00, 16.37, "Albania", "Maize", 36613)

entry2 = (2013, 657.0, 2550.07, 19.76, "Zimbabwe", "Sorghum", 3066)

import pickle

pickle.dump(dtr,open('dtr.pkl','wb'))

pickle.dump(preprocesser,open('preprocessor.pkl','wb'))

import sklearn

print(sklearn.\_\_version\_\_)

from flask import Flask,request, render\_template

import numpy as np

import pickle

import sklearn

print(sklearn.\_\_version\_\_)

#loading models

dtr = pickle.load(open('dtr.pkl','rb'))

preprocessor = pickle.load(open('preprocessor.pkl','rb'))

#flask app

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

@app.route('/')

def index():

    return render\_template('index.html')

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

def predict():

    if request.method == 'POST':

        Year = request.form['Year']

        average\_rain\_fall\_mm\_per\_year = request.form['average\_rain\_fall\_mm\_per\_year']

        pesticides\_tonnes = request.form['pesticides\_tonnes']

        avg\_temp = request.form['avg\_temp']

        Area = request.form['Area']

        Item  = request.form['Item']

        features = np.array([[Year,average\_rain\_fall\_mm\_per\_year,pesticides\_tonnes,avg\_temp,Area,Item]],dtype=object)

        transformed\_features = preprocessor.transform(features)

        prediction = dtr.predict(transformed\_features).reshape(1,-1)

        return render\_template('index.html',prediction = prediction)

if \_\_name\_\_=="\_\_main\_\_":

    app.run(debug=True)

<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Bootstrap demo</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">

  </head>

  <body>

    <h1 class="text-center text-success">Crop Yield Prediction Per Country</h1>

<!--  Year  average\_rain\_fall\_mm\_per\_year pesticides\_tonnes avg\_temp  Area  Item-->

  <div class="container my-4 mt-4" style="background-color: rgba(0, 0, 0, 0.5); border-radius: 20px; color:white">

    <h1 class="text-center text-danger">Input All Features Here</h1>

    <form action="/predict" method="post">

    <div class="form-group">

        <label for="Year">Year</label>

        <input type="number" class="form-control" name="Year" step="any" value="2013">

    </div>

    <div class="form-group">

        <label for="average\_rain\_fall\_mm\_per\_year">average\_rain\_fall\_mm\_per\_year</label>

        <input type="number" class="form-control" name="average\_rain\_fall\_mm\_per\_year" step="any" >

    </div>

    <div class="form-group">

        <label for="pesticides\_tonnes">pesticides\_tonnes</label>

        <input type="number" class="form-control" name="pesticides\_tonnes" step="any" >

    </div>

    <div class="form-group">

        <label for="avg\_temp">avg\_temp</label>

        <input type="number" class="form-control" name="avg\_temp" step="any">

    </div>

    <div class="form-group">

        <label for="Area">Area</label>

        <input type="text" class="form-control" name="Area" >

    </div>

    <div class="form-group">

        <label for="Item">Item</label>

        <input type="text" class="form-control" name="Item" >

    </div>

    <button type="submit" class="btn btn-danger btn-lg mt-2 btn-block">Predict</button>

</form>

      {% if prediction %}

      <h1 class="text-center"> Predicted Yield: <br>{{prediction}}</h1>

      {% endif %}

</div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>

  </body>

</html>

Used **dtr.pkl preprocessor.pkl models**

**Prepare a presentation in proper form for above project for crop yield prediction**