Crop_yield_Prediction

August 29, 2022

[1]:
```
#how to import requirement library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
#how to load the data set
yield_df_data = pd.read_csv("C:\\Users\\Imran\\Desktop\\yield_df.csv")
yield_df_data.head(5)
```

[2]:

	Unnamed: 0	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	1	Albania	Potatoes	1990	66667	1485.0	121.0	16.37
2	2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
3	3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37
4	4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37

[3]:
```
#how to check sha[e of the data set
print("shape of the data set: ", yield_df_data.shape)
```

```
shape of the data set:  (28242, 8)
```

[4]:
```
#how to check nan values
yield_df_data.isnull().sum()
```

[4]:
```
Unnamed: 0                      0
Area                            0
Item                            0
```

```
Year                              0
hg/ha_yield                       0
average_rain_fall_mm_per_year     0
pesticides_tonnes                 0
avg_temp                          0
dtype: int64
```

[5]:
```
#how to check info the data set
yield_df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28242 entries, 0 to 28241
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      28242 non-null  int64
 1   Area                            28242 non-null  object
 2   Item                            28242 non-null  object
 3   Year                            28242 non-null  int64
 4   hg/ha_yield                     28242 non-null  int64
 5   average_rain_fall_mm_per_year   28242 non-null  float64
 6   pesticides_tonnes               28242 non-null  float64
 7   avg_temp                        28242 non-null  float64
dtypes: float64(3), int64(3), object(2)
memory usage: 1.7+ MB
```

[6]:
```
#how to check describe the data set
yield_df_data.describe()
```

[6]:

	Unnamed: 0	Year	hg/ha_yield	\
count	28242.000000	28242.000000	28242.000000	
mean	14120.500000	2001.544296	77053.332094	
std	8152.907488	7.051905	84956.612897	
min	0.000000	1990.000000	50.000000	
25%	7060.250000	1995.000000	19919.250000	
50%	14120.500000	2001.000000	38295.000000	
75%	21180.750000	2008.000000	104676.750000	
max	28241.000000	2013.000000	501412.000000	

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.00000	28242.000000	28242.000000
mean	1149.05598	37076.909344	20.542627
std	709.81215	59958.784665	6.312051
min	51.00000	0.040000	1.300000
25%	593.00000	1702.000000	16.702500
50%	1083.00000	17529.440000	21.510000
75%	1668.00000	48687.880000	26.000000

```
max                          3240.00000       367778.000000      30.650000
```

[7]:
```
#how to check data types
yield_df_data.dtypes
```

[7]:
```
Unnamed: 0                          int64
Area                               object
Item                               object
Year                                int64
hg/ha_yield                         int64
average_rain_fall_mm_per_year     float64
pesticides_tonnes                 float64
avg_temp                          float64
dtype: object
```

[8]:
```
#how to check column
yield_df_data.columns
```

[8]:
```
Index(['Unnamed: 0', 'Area', 'Item', 'Year', 'hg/ha_yield',
       'average_rain_fall_mm_per_year', 'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[9]:
```
#how to check unique value
yield_df_data['Area'].unique()
```

[9]:
```
array(['Albania', 'Algeria', 'Angola', 'Argentina', 'Armenia',
       'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
       'Bangladesh', 'Belarus', 'Belgium', 'Botswana', 'Brazil',
       'Bulgaria', 'Burkina Faso', 'Burundi', 'Cameroon', 'Canada',
       'Central African Republic', 'Chile', 'Colombia', 'Croatia',
       'Denmark', 'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador',
       'Eritrea', 'Estonia', 'Finland', 'France', 'Germany', 'Ghana',
       'Greece', 'Guatemala', 'Guinea', 'Guyana', 'Haiti', 'Honduras',
       'Hungary', 'India', 'Indonesia', 'Iraq', 'Ireland', 'Italy',
       'Jamaica', 'Japan', 'Kazakhstan', 'Kenya', 'Latvia', 'Lebanon',
       'Lesotho', 'Libya', 'Lithuania', 'Madagascar', 'Malawi',
       'Malaysia', 'Mali', 'Mauritania', 'Mauritius', 'Mexico',
       'Montenegro', 'Morocco', 'Mozambique', 'Namibia', 'Nepal',
       'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Norway',
       'Pakistan', 'Papua New Guinea', 'Peru', 'Poland', 'Portugal',
       'Qatar', 'Romania', 'Rwanda', 'Saudi Arabia', 'Senegal',
       'Slovenia', 'South Africa', 'Spain', 'Sri Lanka', 'Sudan',
       'Suriname', 'Sweden', 'Switzerland', 'Tajikistan', 'Thailand',
       'Tunisia', 'Turkey', 'Uganda', 'Ukraine', 'United Kingdom',
       'Uruguay', 'Zambia', 'Zimbabwe'], dtype=object)
```

[5]:
```
#how to check unique value
yield_df_data['Year'].unique()
```

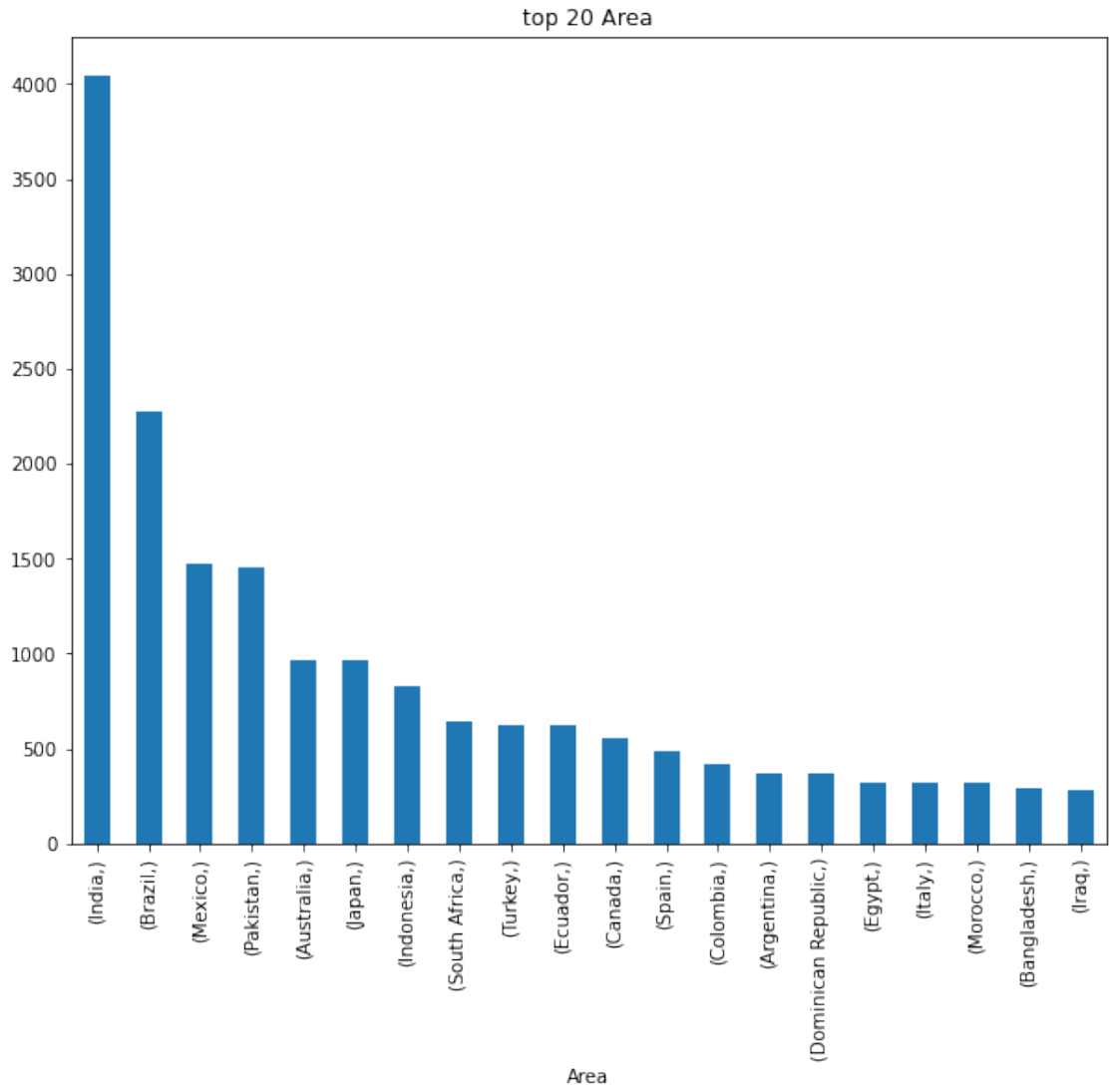
[5]:
```
array([1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
       2001, 2002, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012,
       2013], dtype=int64)
```

[10]:
```
#how to check unique value
yield_df_data['Item'].unique()
```

[10]:
```
array(['Maize', 'Potatoes', 'Rice, paddy', 'Sorghum', 'Soybeans', 'Wheat',
       'Cassava', 'Sweet potatoes', 'Plantains and others', 'Yams'],
      dtype=object)
```

[11]:
```
top_20 = yield_df_data[['Area']].value_counts()[:20]
top_20.plot(kind='bar',figsize=(10,8))
plt.title('top 20 Area')
```

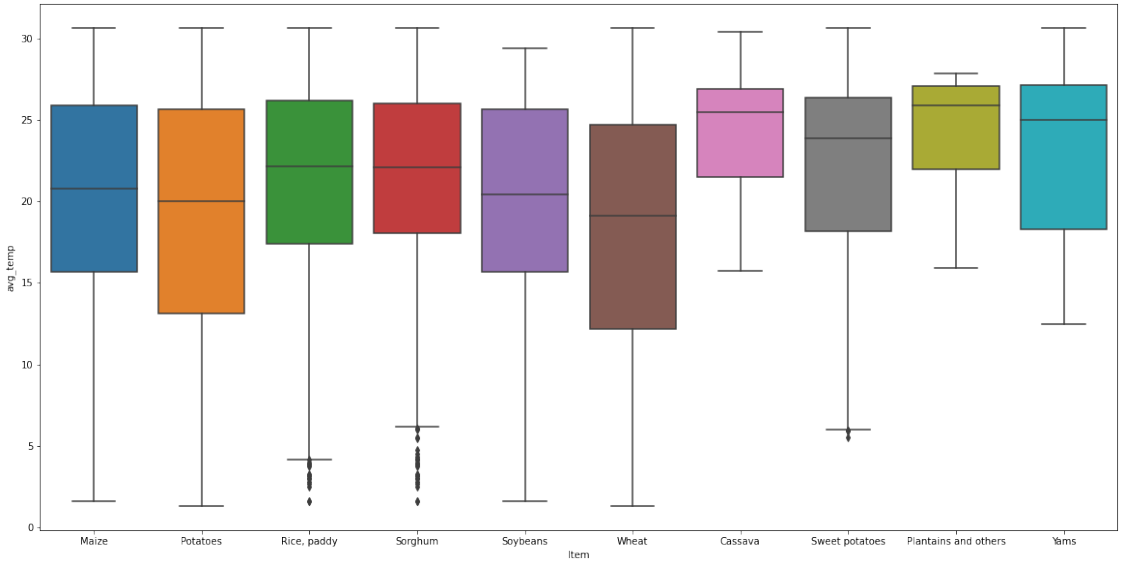
[11]:
```
Text(0.5, 1.0, 'top 20 Area')
```



In the above bar plot India and Brazil are maximum value corresponding to another area.

[12]:
```
# create grouped boxplot
plt.figure(figsize=(20,10))
sns.boxplot(x = yield_df_data['Item'],
            y = yield_df_data['avg_temp'])
```

[12]: <AxesSubplot:xlabel='Item', ylabel='avg_temp'>



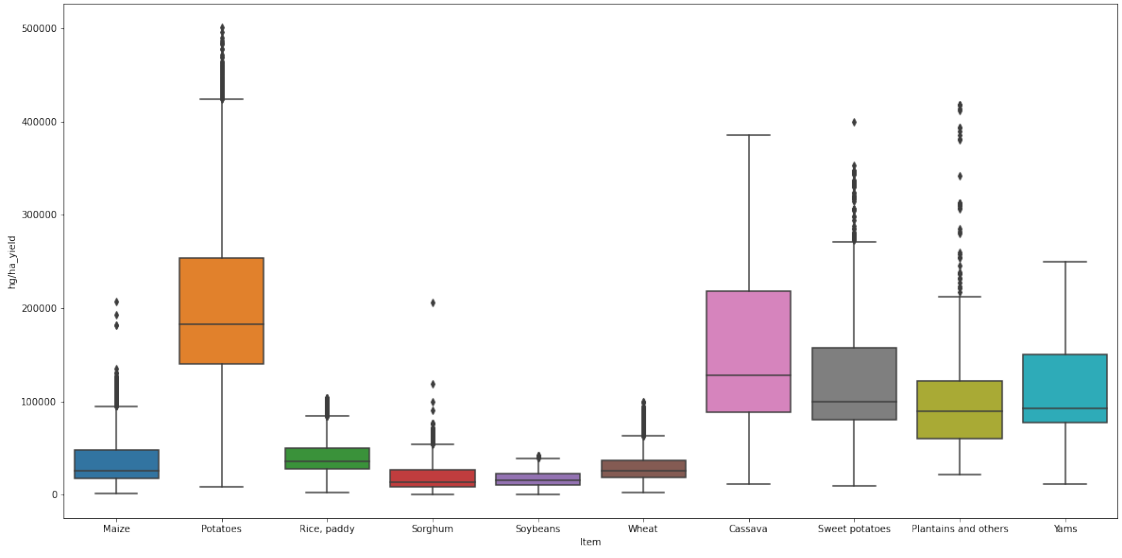
In the Above item like Maize, Potatoes, Rice, paddy, sorghum, wheat, sweet potatoes, Yams is maximum number of avg_tem corresponding to soybeans and Cassava.

[13]:
```
#how to check column
yield_df_data.columns
```

[13]:
```
Index(['Unnamed: 0', 'Area', 'Item', 'Year', 'hg/ha_yield',
       'average_rain_fall_mm_per_year', 'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

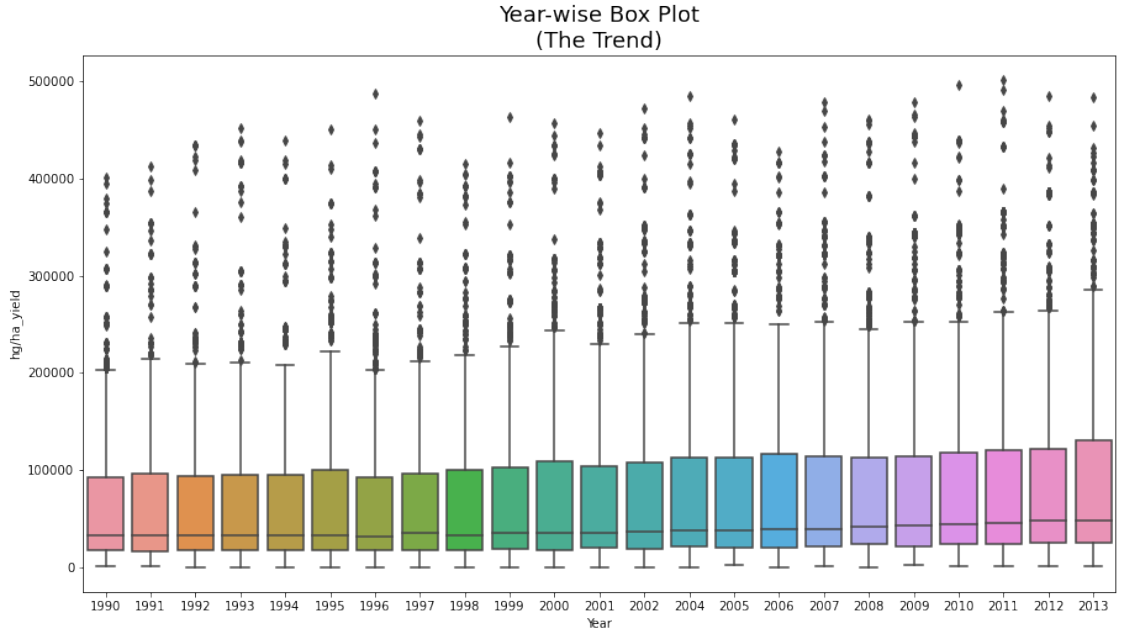
[14]:
```
# create grouped boxplot
plt.figure(figsize=(20,10))
sns.boxplot(x = yield_df_data['Item'],
            y = yield_df_data['hg/ha_yield'])
```

[14]:
```
<AxesSubplot:xlabel='Item', ylabel='hg/ha_yield'>
```



In the above box plot hg/ha yield column corresponding to maximum values in this item like Potatoes, Cassava, Sweet potatoes compare to another box plot.

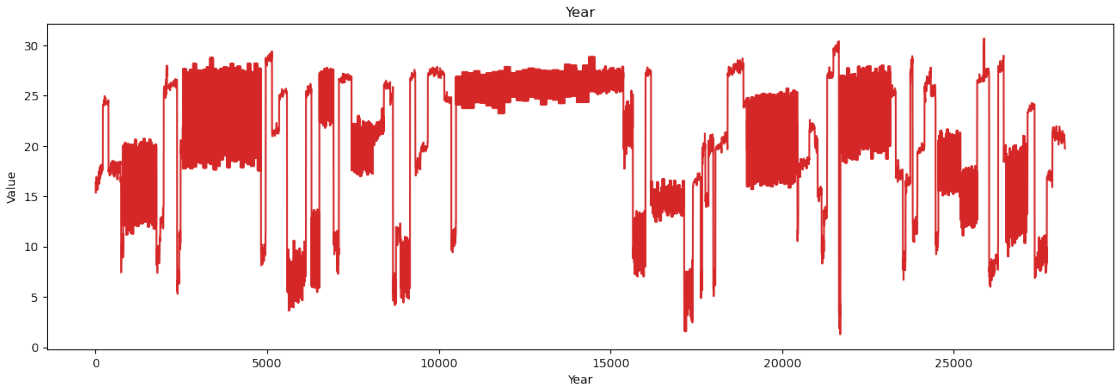
[15]:
```
# Draw Plot
plt.figure(figsize=(15, 8))
sns.boxplot(x='Year', y='hg/ha_yield', data=yield_df_data)
# Set Title
plt.title('Year-wise Box Plot\n(The Trend)', fontsize=18);
```



In the above boxplot the year of 2013 is upward trending corresponding the hg/ha_yield values.

[16]:
```
# Draw Plot
def plot_df(yield_df_data, x, y, title="", xlabel='Year', ylabel='Value',␣
 ↪dpi=100):
    plt.figure(figsize=(16,5), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()

plot_df(yield_df_data, x=yield_df_data.index, y=yield_df_data.avg_temp,␣
 ↪title='Year')
```

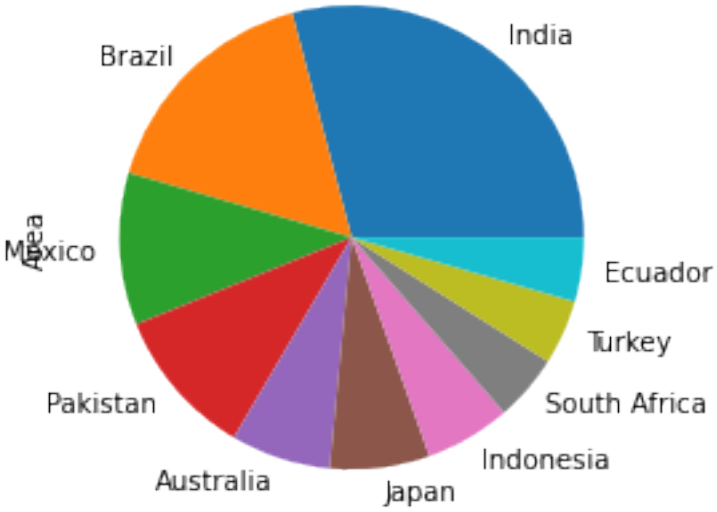


[ ]:
```
```

[17]:
```
yield_df_data.columns
```

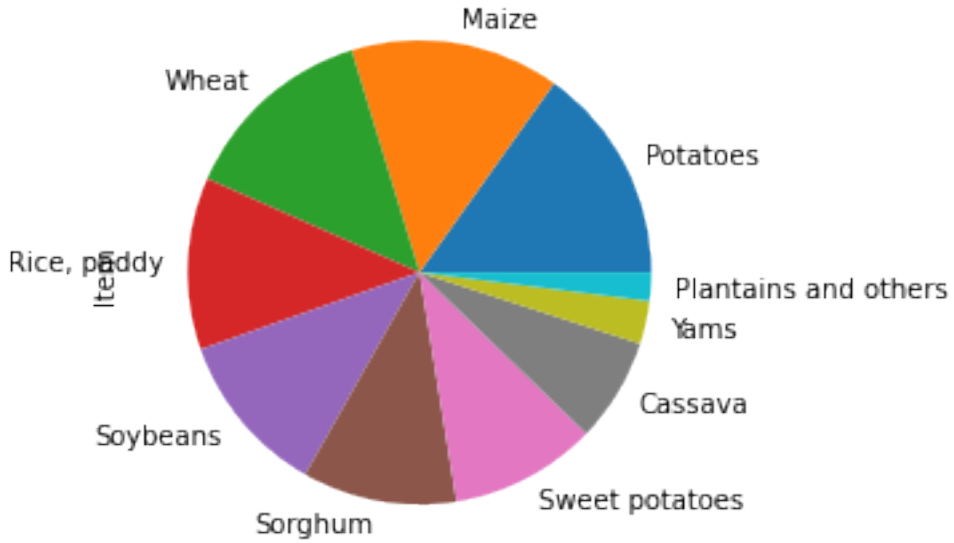
[17]:
```
Index(['Unnamed: 0', 'Area', 'Item', 'Year', 'hg/ha_yield',
       'average_rain_fall_mm_per_year', 'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[18]:
```
yield_df_data['Area'].value_counts()[:10].plot(kind='pie')
plt.show()
```



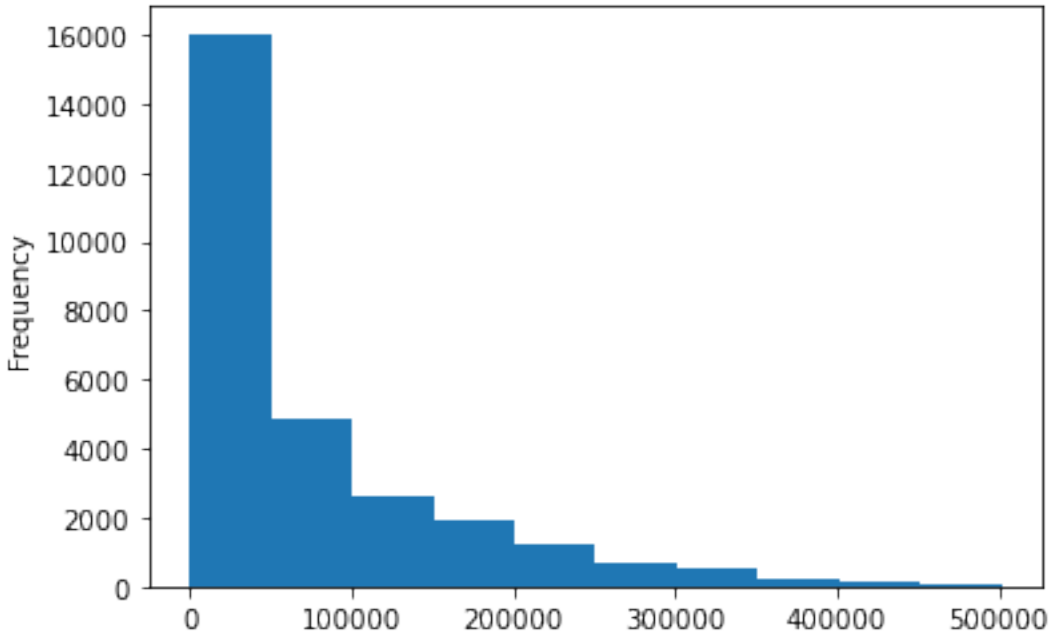
Above pie chart india and Brazil show the heighst area compare to other country.

[19]:
```
yield_df_data['Item'].value_counts()[:10].plot(kind='pie')
plt.show()
```



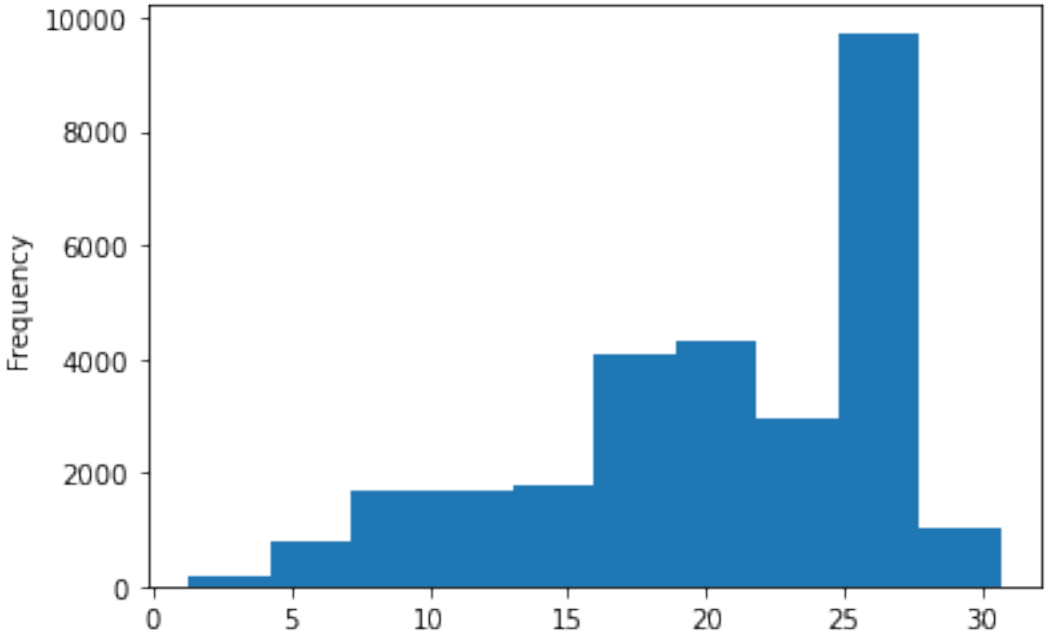
Above pie chart Maize, Potatoes, Wheat show the highest area coresponding to another item.

[20]:
```
yield_df_data['hg/ha_yield'].plot(kind='hist')
plt.show()
```



Above histogram plot 0 to 50000 show the maximum frequency value.

[21]:
```
yield_df_data['avg_temp'].plot(kind='hist')
plt.show()
```



Above histogram plot 25 to 27 show the maximum frequency.

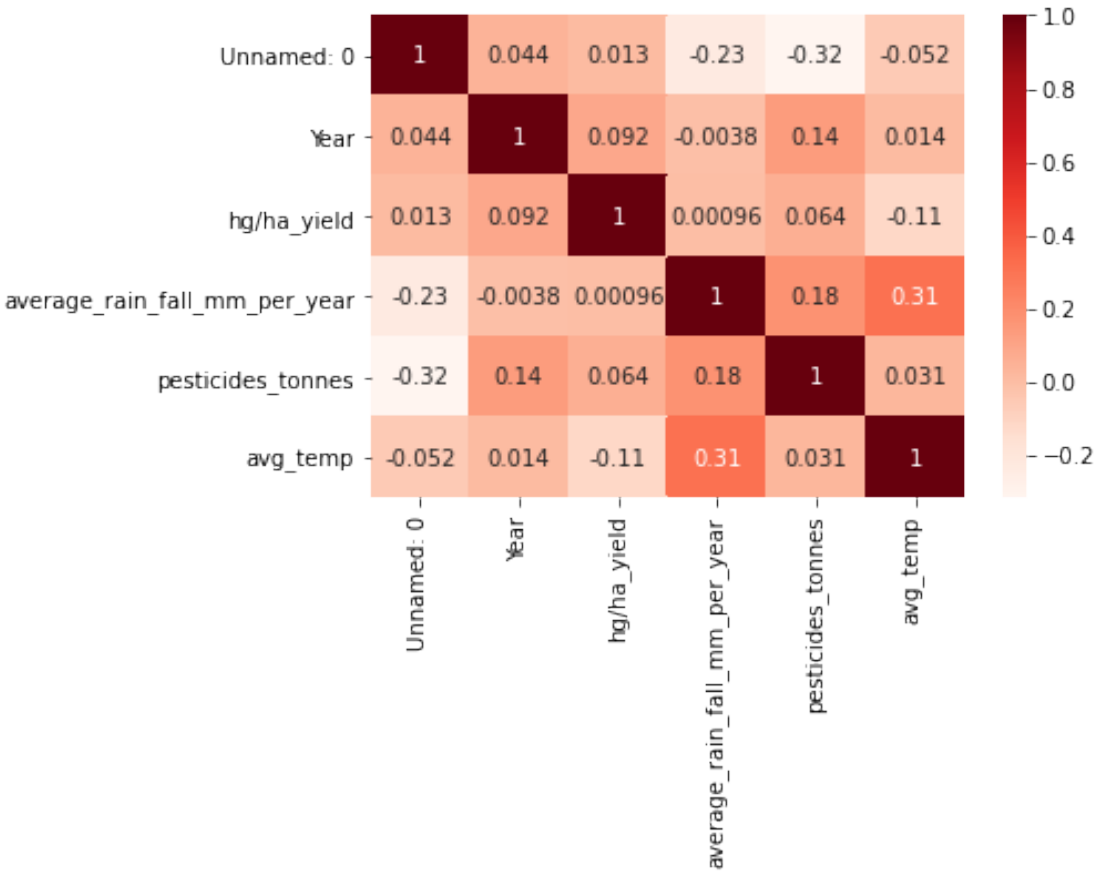
[22]:
```
yield_df_data.corr()
```

[22]:
```
                                Unnamed: 0      Year  hg/ha_yield  \
Unnamed: 0                        1.000000  0.043850     0.013395
Year                              0.043850  1.000000     0.091630
hg/ha_yield                       0.013395  0.091630     1.000000
average_rain_fall_mm_per_year    -0.228755 -0.003798     0.000962
pesticides_tonnes                -0.316017  0.140930     0.064085
avg_temp                         -0.051584  0.014409    -0.114777

                                average_rain_fall_mm_per_year  \
Unnamed: 0                                          -0.228755
Year                                                -0.003798
hg/ha_yield                                          0.000962
average_rain_fall_mm_per_year                        1.000000
pesticides_tonnes                                    0.180984
avg_temp                                             0.313040

                                pesticides_tonnes  avg_temp
Unnamed: 0                          -0.316017 -0.051584
Year                                 0.140930  0.014409
hg/ha_yield                          0.064085 -0.114777
average_rain_fall_mm_per_year        0.180984  0.313040
pesticides_tonnes                    1.000000  0.030946
avg_temp                             0.030946  1.000000
```

[23]:
```
sns.heatmap(yield_df_data.corr(),cmap='Reds',annot=True)
plt.rcParams['figure.figsize']=(10, 5)
```



In the above correlation matrix average_fall_mm_per_year, hg/ha_yield and year are strong correlation to each other compare to another feature.

[24]:
```
#how to remove unnecssary column
yield_df_data.drop(['Unnamed: 0'], axis=1, inplace=True)
```

[25]:
```
#how to check column
yield_df_data.columns
```

[25]:
```
Index(['Area', 'Item', 'Year', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

1 Encoding Categorical Variables

[26]:
```
from sklearn.preprocessing import OneHotEncoder
```

[27]:
```
yield_df_data_onehot = pd.get_dummies(yield_df_data, columns=['Area',"Item"],␣
↪prefix = ['Country',"Item"])
attributes=yield_df_data_onehot.loc[:, yield_df_data_onehot.columns != 'hg/
↪ha_yield']
label=yield_df_data['hg/ha_yield']
attributes.head()
```

[27]:

	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	\
0	1990	1485.0	121.0	16.37	
1	1990	1485.0	121.0	16.37	
2	1990	1485.0	121.0	16.37	
3	1990	1485.0	121.0	16.37	
4	1990	1485.0	121.0	16.37	

	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	\
0	1	0	0	0	
1	1	0	0	0	
2	1	0	0	0	
3	1	0	0	0	
4	1	0	0	0	

	Country_Armenia	Country_Australia	…	Item_Cassava	Item_Maize	\
0	0	0	…	0	1	
1	0	0	…	0	0	
2	0	0	…	0	0	
3	0	0	…	0	0	
4	0	0	…	0	0	

	Item_Plantains and others	Item_Potatoes	Item_Rice, paddy	Item_Sorghum	\
0	0	0	0	0	
1	0	1	0	0	
2	0	0	1	0	
3	0	0	0	1	
4	0	0	0	0	

	Item_Soybeans	Item_Sweet potatoes	Item_Wheat	Item_Yams
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	1	0	0	0

[5 rows x 115 columns]

[28]:
```
attributes.columns
```

[28]:
```
Index(['Year', 'average_rain_fall_mm_per_year', 'pesticides_tonnes',
       'avg_temp', 'Country_Albania', 'Country_Algeria', 'Country_Angola',
       'Country_Argentina', 'Country_Armenia', 'Country_Australia',
       …
       'Item_Cassava', 'Item_Maize', 'Item_Plantains and others',
       'Item_Potatoes', 'Item_Rice, paddy', 'Item_Sorghum', 'Item_Soybeans',
       'Item_Sweet potatoes', 'Item_Wheat', 'Item_Yams'],
      dtype='object', length=115)
```

[29]:
```
attributes = attributes.drop(['Year'], axis=1)
```

[30]:
```
attributes.head()
```

[30]:

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	\
0	1485.0	121.0	16.37	
1	1485.0	121.0	16.37	
2	1485.0	121.0	16.37	
3	1485.0	121.0	16.37	
4	1485.0	121.0	16.37	

	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	\
0	1	0	0	0	
1	1	0	0	0	
2	1	0	0	0	
3	1	0	0	0	
4	1	0	0	0	

	Country_Armenia	Country_Australia	Country_Austria	…	Item_Cassava	\
0	0	0	0	…	0	
1	0	0	0	…	0	
2	0	0	0	…	0	
3	0	0	0	…	0	
4	0	0	0	…	0	

	Item_Maize	Item_Plantains and others	Item_Potatoes	Item_Rice, paddy	\
0	1	0	0	0	
1	0	0	1	0	
2	0	0	0	1	
3	0	0	0	0	
4	0	0	0	0	

	Item_Sorghum	Item_Soybeans	Item_Sweet potatoes	Item_Wheat	Item_Yams
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	1	0	0	0	0
4	0	1	0	0	0

```
[5 rows x 114 columns]
```

[31]:
```
#Scaling Features
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
attributes=scaler.fit_transform(attributes)
```

[32]:
```
attributes
```

[32]:
```
array([[4.49670743e-01, 3.28894097e-04, 5.13458262e-01, …,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.49670743e-01, 3.28894097e-04, 5.13458262e-01, …,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.49670743e-01, 3.28894097e-04, 5.13458262e-01, …,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       …,
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, …,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, …,
        1.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.90028222e-01, 6.93361288e-03, 6.28960818e-01, …,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

[33]:
```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(attributes, label,␣
 ↪test_size=0.2, random_state=42)
```

[34]:
```
# import the regressor
from sklearn.ensemble import RandomForestRegressor
```

[35]:
```
regressor = RandomForestRegressor()
```

[36]:
```
regressor.fit(train_x, train_y)
```

[36]:
```
RandomForestRegressor()
```

[37]:
```
y_pred = regressor.predict(test_x)
```

[38]:
```
y_pred
```

[38]:
```
array([ 71119.18,  23816.02,  53240.77, …, 252775.  ,  24848.4 ,
        23109.18])
```

[39]:
```
from sklearn.metrics import r2_score
score = r2_score(test_y,y_pred)
score
```

[39]:
```
0.9741897720376581
```

[40]:
```
# Calculating the MSE with sklearn
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(test_y, y_pred)
print(mse)
```

```
187218927.13208273
```

[6]:
```
yield_df_data.columns
```

[6]:
```
Index(['Unnamed: 0', 'Area', 'Item', 'Year', 'hg/ha_yield',
       'average_rain_fall_mm_per_year', 'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[ ]:
```

```

[1]:
```
#how to load the data set
import pandas as pd
yield_df_data = pd.read_csv("C:\\Users\\Imran\\Desktop\\yield_df.csv",␣
↪header=0, index_col=0)
yield_df_data.head(5)
```

[1]:

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	Albania	Potatoes	1990	66667	1485.0	121.0	16.37
2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37
4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37

[2]:
```
yield_df_data.columns
```

[2]:
```
Index(['Area', 'Item', 'Year', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[3]:
```
yield_df_data = yield_df_data[['Year', 'Area', 'Item', 'hg/ha_yield',␣
↪'average_rain_fall_mm_per_year','pesticides_tonnes', 'avg_temp']]
```

[4]:
```
yield_df_data
```

[4]:
```
       Year      Area            Item  hg/ha_yield  \
0      1990   Albania           Maize        36613
1      1990   Albania        Potatoes        66667
2      1990   Albania     Rice, paddy        23333
3      1990   Albania         Sorghum        12500
4      1990   Albania        Soybeans         7000
…       …        …              …           …
28237  2013  Zimbabwe     Rice, paddy        22581
28238  2013  Zimbabwe         Sorghum         3066
28239  2013  Zimbabwe        Soybeans        13142
28240  2013  Zimbabwe  Sweet potatoes        22222
28241  2013  Zimbabwe           Wheat        22888

       average_rain_fall_mm_per_year  pesticides_tonnes  avg_temp
0                             1485.0             121.00     16.37
1                             1485.0             121.00     16.37
2                             1485.0             121.00     16.37
3                             1485.0             121.00     16.37
4                             1485.0             121.00     16.37
…                              …                 …        …
28237                          657.0            2550.07     19.76
28238                          657.0            2550.07     19.76
28239                          657.0            2550.07     19.76
28240                          657.0            2550.07     19.76
28241                          657.0            2550.07     19.76

[28242 rows x 7 columns]
```

[5]:
```
yield_df_data['Area'] = (yield_df_data['Area']+yield_df_data['Area']).str.
 ↪lower()
yield_df_data['Area'] = yield_df_data['Area'].str.lower()
yield_df_data['Area'] = yield_df_data['Area'].str.lower()
yield_df_data.dropna(inplace=True)
d = yield_df_data
```

[6]:
```
d['Area'].unique()
```

[6]:
```
array(['albaniaalbania', 'algeriaalgeria', 'angolaangola',
       'argentinaargentina', 'armeniaarmenia', 'australiaaustralia',
       'austriaaustria', 'azerbaijanazerbaijan', 'bahamasbahamas',
       'bahrainbahrain', 'bangladeshbangladesh', 'belarusbelarus',
       'belgiumbelgium', 'botswanabotswana', 'brazilbrazil',
       'bulgariabulgaria', 'burkina fasoburkina faso', 'burundiburundi',
       'camerooncameroon', 'canadacanada',
       'central african republiccentral african republic', 'chilechile',
       'colombiacolombia', 'croatiacroatia', 'denmarkdenmark',
       'dominican republicdominican republic', 'ecuadorecuador',
```

```
       'egyptegypt', 'el salvadorel salvador', 'eritreaeritrea',
       'estoniaestonia', 'finlandfinland', 'francefrance',
       'germanygermany', 'ghanaghana', 'greecegreece',
       'guatemalaguatemala', 'guineaguinea', 'guyanaguyana', 'haitihaiti',
       'hondurashonduras', 'hungaryhungary', 'indiaindia',
       'indonesiaindonesia', 'iraqiraq', 'irelandireland', 'italyitaly',
       'jamaicajamaica', 'japanjapan', 'kazakhstankazakhstan',
       'kenyakenya', 'latvialatvia', 'lebanonlebanon', 'lesotholesotho',
       'libyalibya', 'lithuanialithuania', 'madagascarmadagascar',
       'malawimalawi', 'malaysiamalaysia', 'malimali',
       'mauritaniamauritania', 'mauritiusmauritius', 'mexicomexico',
       'montenegromontenegro', 'moroccomorocco', 'mozambiquemozambique',
       'namibianamibia', 'nepalnepal', 'netherlandsnetherlands',
       'new zealandnew zealand', 'nicaraguanicaragua', 'nigerniger',
       'norwaynorway', 'pakistanpakistan',
       'papua new guineapapua new guinea', 'peruperu', 'polandpoland',
       'portugalportugal', 'qatarqatar', 'romaniaromania', 'rwandarwanda',
       'saudi arabiasaudi arabia', 'senegalsenegal', 'sloveniaslovenia',
       'south africasouth africa', 'spainspain', 'sri lankasri lanka',
       'sudansudan', 'surinamesuriname', 'swedensweden',
       'switzerlandswitzerland', 'tajikistantajikistan',
       'thailandthailand', 'tunisiatunisia', 'turkeyturkey',
       'ugandauganda', 'ukraineukraine', 'united kingdomunited kingdom',
       'uruguayuruguay', 'zambiazambia', 'zimbabwezimbabwe'], dtype=object)
```

[7]:
```
country = ['albaniaalbania', 'algeriaalgeria', 'angolaangola',
       'argentinaargentina', 'armeniaarmenia', 'australiaaustralia',
       'austriaaustria', 'azerbaijanazerbaijan', 'bahamasbahamas',
       'bahrainbahrain', 'bangladeshbangladesh', 'belarusbelarus',
       'belgiumbelgium', 'botswanabotswana', 'brazilbrazil',
       'bulgariabulgaria', 'burkina fasoburkina faso', 'burundiburundi',
       'camerooncameroon', 'canadacanada',
       'central african republiccentral african republic', 'chilechile',
       'colombiacolombia', 'croatiacroatia', 'denmarkdenmark',
       'dominican republicdominican republic', 'ecuadorecuador',
       'egyptegypt', 'el salvadorel salvador', 'eritreaeritrea',
       'estoniaestonia', 'finlandfinland', 'francefrance',
       'germanygermany', 'ghanaghana', 'greecegreece',
       'guatemalaguatemala', 'guineaguinea', 'guyanaguyana', 'haitihaiti',
       'hondurashonduras', 'hungaryhungary', 'indiaindia',
       'indonesiaindonesia', 'iraqiraq', 'irelandireland', 'italyitaly',
       'jamaicajamaica', 'japanjapan', 'kazakhstankazakhstan',
       'kenyakenya', 'latvialatvia', 'lebanonlebanon', 'lesotholesotho',
       'libyalibya', 'lithuanialithuania', 'madagascarmadagascar',
       'malawimalawi', 'malaysiamalaysia', 'malimali',
       'mauritaniamauritania', 'mauritiusmauritius', 'mexicomexico',
       'montenegromontenegro', 'moroccomorocco', 'mozambiquemozambique',
```

```
       'namibianamibia', 'nepalnepal', 'netherlandsnetherlands',
       'new zealandnew zealand', 'nicaraguanicaragua', 'nigerniger',
       'norwaynorway', 'pakistanpakistan',
       'papua new guineapapua new guinea', 'peruperu', 'polandpoland',
       'portugalportugal', 'qatarqatar', 'romaniaromania', 'rwandarwanda',
       'saudi arabiasaudi arabia', 'senegalsenegal', 'sloveniaslovenia',
       'south africasouth africa', 'spainspain', 'sri lankasri lanka',
       'sudansudan', 'surinamesuriname', 'swedensweden',
       'switzerlandswitzerland', 'tajikistantajikistan',
       'thailandthailand', 'tunisiatunisia', 'turkeyturkey',
       'ugandauganda', 'ukraineukraine', 'united kingdomunited kingdom',
       'uruguayuruguay', 'zambiazambia', 'zimbabwezimbabwe']
```

[8]:
```
for cont in country:
    if cont in yield_df_data['Area'].unique():
        print(cont)
```

```
albaniaalbania
algeriaalgeria
angolaangola
argentinaargentina
armeniaarmenia
australiaaustralia
austriaaustria
azerbaijanazerbaijan
bahamasbahamas
bahrainbahrain
bangladeshbangladesh
belarusbelarus
belgiumbelgium
botswanabotswana
brazilbrazil
bulgariabulgaria
burkina fasoburkina faso
burundiburundi
camerooncameroon
canadacanada
central african republiccentral african republic
chilechile
colombiacolombia
croatiacroatia
denmarkdenmark
dominican republicdominican republic
ecuadorecuador
egyptegypt
el salvadorel salvador
eritreaeritrea
```

estoniaestonia
finlandfinland
francefrance
germanygermany
ghanaghana
greecegreece
guatemalaguatemala
guineaguinea
guyanaguyana
haitihaiti
hondurashonduras
hungaryhungary
indiaindia
indonesiaindonesia
iraqiraq
irelandireland
italyitaly
jamaicajamaica
japanjapan
kazakhstankazakhstan
kenyakenya
latvialatvia
lebanonlebanon
lesotholesotho
libyalibya
lithuanialithuania
madagascarmadagascar
malawimalawi
malaysiamalaysia
malimali
mauritaniamauritania
mauritiusmauritius
mexicomexico
montenegromontenegro
moroccomorocco
mozambiquemozambique
namibianamibia
nepalnepal
netherlandsnetherlands
new zealandnew zealand
nicaraguanicaragua
nigerniger
norwaynorway
pakistanpakistan
papua new guineapapua new guinea
peruperu
polandpoland
portugalportugal

```
qatarqatar
romaniaromania
rwandarwanda
saudi arabiasaudi arabia
senegalsenegal
sloveniaslovenia
south africasouth africa
spainspain
sri lankasri lanka
sudansudan
surinamesuriname
swedensweden
switzerlandswitzerland
tajikistantajikistan
thailandthailand
tunisiatunisia
turkeyturkey
ugandauganda
ukraineukraine
united kingdomunited kingdom
uruguayuruguay
zambiazambia
zimbabwezimbabwe
```

[9]:
```
combined_data = pd.DataFrame()
for cont in country:
    data = yield_df_data[yield_df_data['Area']==cont]

    combined_data = combined_data.append(data)

combined_data
```

[9]:

	Year	Area	Item	hg/ha_yield	\
0	1990	albaniaalbania	Maize	36613	
1	1990	albaniaalbania	Potatoes	66667	
2	1990	albaniaalbania	Rice, paddy	23333	
3	1990	albaniaalbania	Sorghum	12500	
4	1990	albaniaalbania	Soybeans	7000	
…	…	…	…	…	
28237	2013	zimbabwezimbabwe	Rice, paddy	22581	
28238	2013	zimbabwezimbabwe	Sorghum	3066	
28239	2013	zimbabwezimbabwe	Soybeans	13142	
28240	2013	zimbabwezimbabwe	Sweet potatoes	22222	
28241	2013	zimbabwezimbabwe	Wheat	22888	

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	1485.0	121.00	16.37

```
1                              1485.0              121.00     16.37
2                              1485.0              121.00     16.37
3                              1485.0              121.00     16.37
4                              1485.0              121.00     16.37
…                               …                  …       …
28237                           657.0             2550.07     19.76
28238                           657.0             2550.07     19.76
28239                           657.0             2550.07     19.76
28240                           657.0             2550.07     19.76
28241                           657.0             2550.07     19.76

[28242 rows x 7 columns]
```

[10]:
```
series = combined_data
```

[11]:
```
series
```

[11]:
```
       Year              Area            Item  hg/ha_yield  \
0      1990    albaniaalbania           Maize        36613
1      1990    albaniaalbania        Potatoes        66667
2      1990    albaniaalbania     Rice, paddy        23333
3      1990    albaniaalbania         Sorghum        12500
4      1990    albaniaalbania        Soybeans         7000
…     …                 …              …           …
28237  2013  zimbabwezimbabwe     Rice, paddy        22581
28238  2013  zimbabwezimbabwe         Sorghum         3066
28239  2013  zimbabwezimbabwe        Soybeans        13142
28240  2013  zimbabwezimbabwe  Sweet potatoes        22222
28241  2013  zimbabwezimbabwe           Wheat        22888

       average_rain_fall_mm_per_year  pesticides_tonnes  avg_temp
0                             1485.0             121.00     16.37
1                             1485.0             121.00     16.37
2                             1485.0             121.00     16.37
3                             1485.0             121.00     16.37
4                             1485.0             121.00     16.37
…                              …                 …       …
28237                          657.0            2550.07     19.76
28238                          657.0            2550.07     19.76
28239                          657.0            2550.07     19.76
28240                          657.0            2550.07     19.76
28241                          657.0            2550.07     19.76

[28242 rows x 7 columns]
```

[12]:
```
series.columns
```

[12]:
```
Index(['Year', 'Area', 'Item', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[13]:
```
# series.drop(['Item'], axis=1, inplace=True)
```

[14]:
```
# series.drop(['Area'], axis=1, inplace=True)
```

[15]:
```
series.columns
```

[15]:
```
Index(['Year', 'Area', 'Item', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[16]:
```
# #how to drop values
# series.drop(['Year'], axis=1, inplace=True)
```

[17]:
```
series.columns
```

[17]:
```
Index(['Year', 'Area', 'Item', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[18]:
```
# last_year = series[['Year']]
```

[19]:
```
# last_year
```

[20]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

series['Area'] = le.fit_transform(series['Area'])
series['Item'] = le.fit_transform(series['Item'])

print(series.tail())
values = series.values
```

```
       Year  Area  Item  hg/ha_yield  average_rain_fall_mm_per_year  \
28237  2013   100     4        22581                          657.0
28238  2013   100     5         3066                          657.0
28239  2013   100     6        13142                          657.0
28240  2013   100     7        22222                          657.0
28241  2013   100     8        22888                          657.0

       pesticides_tonnes  avg_temp
28237            2550.07     19.76
28238            2550.07     19.76
28239            2550.07     19.76
```

```
28240            2550.07     19.76
28241            2550.07     19.76
```

[21]:
```
# transform a time series dataset into a supervised learning dataset
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
    agg = pd.concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

[22]:
```
# transform the time series data into supervised learning
data = series_to_supervised(values, n_in=2)
```

[23]:
```
data
```

[23]:
```
array([[1.99000e+03, 0.00000e+00, 1.00000e+00, …, 1.48500e+03,
        1.21000e+02, 1.63700e+01],
       [1.99000e+03, 0.00000e+00, 3.00000e+00, …, 1.48500e+03,
        1.21000e+02, 1.63700e+01],
       [1.99000e+03, 0.00000e+00, 4.00000e+00, …, 1.48500e+03,
        1.21000e+02, 1.63700e+01],
       …,
       [2.01300e+03, 1.00000e+02, 4.00000e+00, …, 6.57000e+02,
        2.55007e+03, 1.97600e+01],
       [2.01300e+03, 1.00000e+02, 5.00000e+00, …, 6.57000e+02,
        2.55007e+03, 1.97600e+01],
       [2.01300e+03, 1.00000e+02, 6.00000e+00, …, 6.57000e+02,
        2.55007e+03, 1.97600e+01]])
```

[24]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data)
trainX, trainy = train[:, 1:], train[:, 0]
testX, testy = test[:, 1:], test[:, 0]
```

[25]:
```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100)
```

```
model.fit(trainX, trainy)
yhat = model.predict(testX)
```

[26]:
```
yhat
```

[26]:
```
array([2008.  , 2004.  , 1990.  , …, 2004.  , 2008.98, 1997.  ])
```

[27]:
```
import pickle
```

[28]:
```
saved_model = pickle.dump(model, open("C:\\Users\\Imran\\Desktop\\model.pkl",␣
↪"wb"))
```

[29]:
```
#loading save model
rf_saved_model = pickle.load(open("C:\\Users\\Imran\\Desktop\\model.pkl", "rb"))
```

[30]:
```
print("Please choose among these country and city only for temperature␣
↪prediction")
for cont in country:
    if cont in d['Area'].unique():
        print(cont, d[d['Area']==cont]['Item'].unique())
        print()
```

```
Please choose among these country and city only for temperature prediction
albaniaalbania ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

algeriaalgeria ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Wheat']

angolaangola ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sweet potatoes'
'Wheat'
 'Sorghum' 'Soybeans']

argentinaargentina ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum'
'Soybeans'
 'Sweet potatoes' 'Wheat']

armeniaarmenia ['Maize' 'Potatoes' 'Wheat']

australiaaustralia ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

austriaaustria ['Maize' 'Potatoes' 'Soybeans' 'Wheat' 'Sorghum']

azerbaijanazerbaijan ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
'Wheat']

bahamasbahamas ['Cassava' 'Maize' 'Sweet potatoes' 'Plantains and others']
```

```
bahrainbahrain ['Potatoes' 'Sweet potatoes']

bangladeshbangladesh ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Sweet
potatoes' 'Wheat'
 'Soybeans']

belarusbelarus ['Maize' 'Potatoes' 'Wheat']

belgiumbelgium ['Maize' 'Potatoes' 'Wheat']

botswanabotswana ['Maize' 'Sorghum' 'Wheat']

brazilbrazil ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat' 'Yams']

bulgariabulgaria ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

burkina fasoburkina faso ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum'
'Soybeans'
 'Sweet potatoes' 'Yams']

burundiburundi ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat' 'Yams']

cameroooncameroon ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes' 'Wheat' 'Yams']

canadacanada ['Maize' 'Potatoes' 'Soybeans' 'Wheat']

central african republiccentral african republic ['Cassava' 'Maize' 'Plantains
and others' 'Potatoes' 'Rice, paddy'
 'Sorghum' 'Yams']

chilechile ['Maize' 'Potatoes' 'Rice, paddy' 'Sweet potatoes' 'Wheat']

colombiacolombia ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Soybeans' 'Wheat' 'Yams']

croatiacroatia ['Maize' 'Potatoes' 'Sorghum' 'Soybeans' 'Wheat']

denmarkdenmark ['Maize' 'Potatoes' 'Wheat']

dominican republicdominican republic ['Cassava' 'Maize' 'Plantains and others'
'Potatoes' 'Rice, paddy'
 'Sorghum' 'Sweet potatoes' 'Yams']
```

```
ecuadorecuador ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes' 'Wheat']

egyptegypt ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

el salvadorel salvador ['Cassava' 'Maize' 'Plantains and others' 'Potatoes'
'Rice, paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes']

eritreaeritrea ['Maize' 'Potatoes' 'Sorghum' 'Wheat']

estoniaestonia ['Potatoes' 'Wheat']

finlandfinland ['Potatoes' 'Wheat']

francefrance ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

germanygermany ['Maize' 'Potatoes' 'Soybeans' 'Wheat']

ghanaghana ['Cassava' 'Maize' 'Plantains and others' 'Rice, paddy' 'Sorghum'
'Yams'
 'Sweet potatoes']

greecegreece ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

guatemalaguatemala ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Soybeans' 'Wheat' 'Sweet potatoes']

guineaguinea ['Cassava' 'Maize' 'Plantains and others' 'Rice, paddy' 'Sorghum'
 'Sweet potatoes' 'Yams' 'Potatoes']

guyanaguyana ['Cassava' 'Maize' 'Plantains and others' 'Rice, paddy' 'Sweet
potatoes'
 'Yams']

haitihaiti ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice, paddy'
 'Sorghum' 'Sweet potatoes' 'Yams']

hondurashonduras ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Sweet potatoes' 'Wheat' 'Soybeans']
```

```
hungaryhungary ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

indiaindia ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat']

indonesiaindonesia ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Soybeans' 'Sweet
potatoes']

iraqiraq ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

irelandireland ['Potatoes' 'Wheat']

italyitaly ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

jamaicajamaica ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sweet potatoes' 'Yams']

japanjapan ['Maize' 'Potatoes' 'Rice, paddy' 'Soybeans' 'Sweet potatoes' 'Wheat'
 'Yams']

kazakhstankazakhstan ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
'Wheat']

kenyakenya ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice, paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes' 'Wheat' 'Yams']

latvialatvia ['Potatoes' 'Wheat']

lebanonlebanon ['Maize' 'Potatoes' 'Sorghum' 'Wheat']

lesotholesotho ['Maize' 'Potatoes' 'Sorghum' 'Wheat']

libyalibya ['Maize' 'Potatoes' 'Wheat']

lithuanialithuania ['Maize' 'Potatoes' 'Wheat']

madagascarmadagascar ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum'
'Soybeans'
 'Sweet potatoes' 'Wheat']

malawimalawi ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice, paddy'
 'Sorghum' 'Wheat' 'Soybeans']

malaysiamalaysia ['Cassava' 'Maize' 'Rice, paddy' 'Soybeans' 'Sweet potatoes']
```

```
malimali ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat' 'Yams']

mauritaniamauritania ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Sweet
potatoes' 'Wheat'
 'Yams']

mauritiusmauritius ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sweet potatoes']

mexicomexico ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat']

montenegromontenegro ['Maize' 'Potatoes' 'Wheat']

moroccomorocco ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

mozambiquemozambique ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum'
'Sweet potatoes'
 'Wheat']

namibianamibia ['Maize' 'Sorghum' 'Wheat' 'Potatoes']

nepalnepal ['Maize' 'Potatoes' 'Rice, paddy' 'Soybeans' 'Wheat']

netherlandsnetherlands ['Maize' 'Potatoes' 'Wheat']

new zealandnew zealand ['Maize' 'Potatoes' 'Sweet potatoes' 'Wheat']

nicaraguanicaragua ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Soybeans' 'Yams']

nigerniger ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Sweet
potatoes'
 'Wheat']

norwaynorway ['Potatoes' 'Wheat']

pakistanpakistan ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

papua new guineapapua new guinea ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy'
'Sorghum' 'Sweet potatoes'
 'Yams']
```

```
peruperu ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice, paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes' 'Wheat']

polandpoland ['Maize' 'Potatoes' 'Wheat' 'Soybeans']

portugalportugal ['Maize' 'Potatoes' 'Rice, paddy' 'Sweet potatoes' 'Wheat'
'Yams']

qatarqatar ['Maize' 'Potatoes' 'Wheat']

romaniaromania ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

rwandarwanda ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat' 'Yams']

saudi arabiasaudi arabia ['Maize' 'Potatoes' 'Sorghum' 'Wheat']

senegalsenegal ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Sweet
potatoes']

sloveniaslovenia ['Maize' 'Potatoes' 'Soybeans' 'Wheat']

south africasouth africa ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
'Sweet potatoes'
 'Wheat']

spainspain ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

sri lankasri lanka ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice,
paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes']

sudansudan ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Sweet potatoes' 'Wheat'
 'Yams']

surinamesuriname ['Cassava' 'Maize' 'Plantains and others' 'Rice, paddy'
'Soybeans'
 'Sweet potatoes']

swedensweden ['Potatoes' 'Wheat']

switzerlandswitzerland ['Maize' 'Potatoes' 'Soybeans' 'Wheat']

tajikistantajikistan ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
'Wheat']
```

```
thailandthailand ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum'
'Soybeans' 'Wheat']

tunisiatunisia ['Potatoes' 'Sorghum' 'Wheat']

turkeyturkey ['Maize' 'Potatoes' 'Rice, paddy' 'Soybeans' 'Wheat' 'Sorghum']

ugandauganda ['Cassava' 'Maize' 'Plantains and others' 'Potatoes' 'Rice, paddy'
 'Sorghum' 'Soybeans' 'Sweet potatoes' 'Wheat']

ukraineukraine ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Wheat']

united kingdomunited kingdom ['Potatoes' 'Wheat']

uruguayuruguay ['Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans' 'Sweet
potatoes'
 'Wheat']

zambiazambia ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum' 'Soybeans'
 'Sweet potatoes' 'Wheat']

zimbabwezimbabwe ['Cassava' 'Maize' 'Potatoes' 'Rice, paddy' 'Sorghum'
'Soybeans'
 'Sweet potatoes' 'Wheat']
```

[31]:
```
item = input("Enter the name of the item name :").lower()
Area = input( "Enter the name of Country : ").lower()
year = input("Enter the year :")
y = int(year)
```

```
Enter the name of the item name :Maize
Enter the name of Country : albaniaalbania
Enter the year :2013
```

[67]:
```
######################## creating INDEX for input month and year
######################## extracting previous as well as next years data for␣
 ↪supervised learning purpose
# test = pd.DataFrame(columns  =['Year', 'Area', 'Item', 'hg/ha_yield'])
# test = pd.DataFrame()
# flag = 0
# if y<=2012:
#   ind = year
#   test = pd.DataFrame()
#   for year in (y-1,y,y+1):
#     year = str(year)
```

```
#     for i in range(1,13):
#       if i<10:
#         date = year+'-0'+str(i)+'-01'
#       else:
#         date = year+'-'+str(i)+'-01'
#       same_dates = d.loc[date]
#       # intermediate = same_dates[(same_dates['City']==city) &␣
 ↪(same_dates['Country']==country)]
#       # if intermediate.shape[0]>1:
#       #   print(intermediate.groupby('dt').mean())
#       test = test.append(same_dates[(same_dates['Year']==city) &␣
 ↪(same_dates['Year']==country)],ignore_index=True)



# else:
#   flag = 1
#   ind = year
#   for year in range(2013,y+2):
#     year_art = str(2012-(year-2013))
#     year = str(year)

#     for i in range(1,13):
#       if i<10:
#         date = year+'-0'+str(i)+'-01'
#         date_art = year_art+'-0'+str(i)+'-01'

#       else:
#         date = year+'-'+str(i)+'-01'
#         date_art = year_art+'-'+str(i)+'-01'

#       same_dates = d.loc[date_art]
#       test = test.append(same_dates[(same_dates['Year']==city) &␣
 ↪(same_dates['Year']==country)],ignore_index=True)
      #test = test.append({'dt': date, 'AverageTemperature': 0, 'City_Country':␣
 ↪city+country},ignore_index=True)
  #test.set_index('dt', inplace=True)
#test.
 ↪drop(['City','Country','AverageTemperatureUncertainty','Latitude','Longitude'],␣
 ↪axis = 1, inplace = True)
#test.drop_duplicates(inplace = True)
#print(test)
```

[43]:
```
xtest = series_to_supervised(values,n_in=2)
```

[44]:
```
testX, testy = xtest[:, 1:], xtest[:, 3]
```

[45]:
```
testy
```

[45]:
```
array([36613., 66667., 23333., …, 22581.,  3066., 13142.])
```

[46]:
```
y_pred = rf_saved_model.predict(testX)
```

[47]:
```
from sklearn.metrics import mean_absolute_error
```

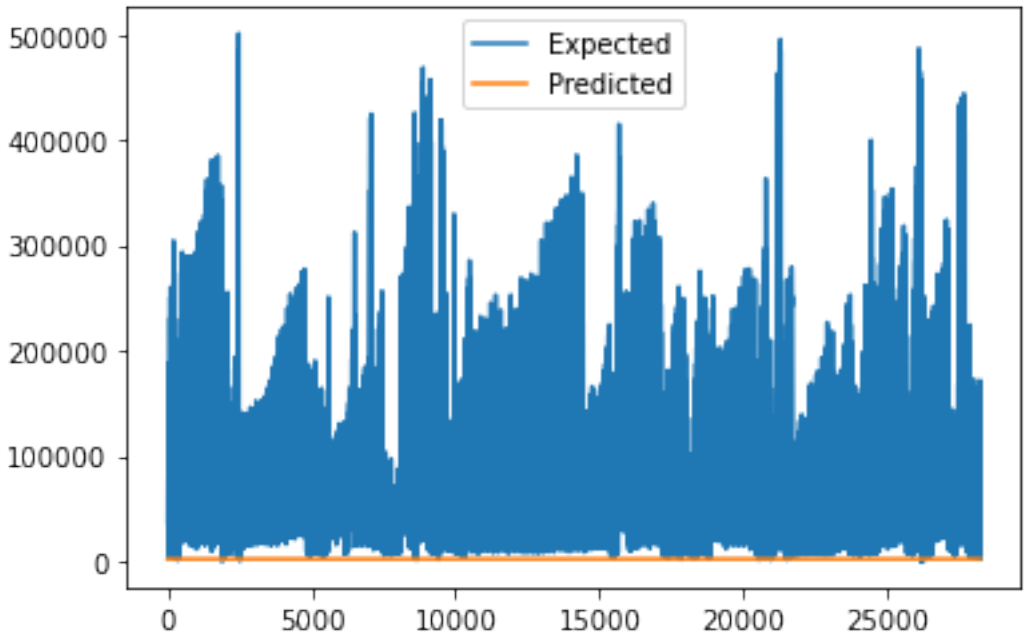
[48]:
```
mean_absolute_error(testy, y_pred)
```

[48]:
```
75059.93031586403
```

[49]:
```
from sklearn.metrics import r2_score
score = r2_score(testy, y_pred)
score
```

[49]:
```
-0.7804790740172789
```

[69]:
```
# plot expected vs predicted
from matplotlib import pyplot
import random
pyplot.plot(testy, label='Expected')
pyplot.plot(y_pred, label='Predicted')
pyplot.legend()
pyplot.show()
```



[71]:
```
# # forecast monthly births with random forest
# from numpy import asarray
# from pandas import read_csv
# from pandas import DataFrame
# from pandas import concat
# from sklearn.metrics import mean_absolute_error
# from sklearn.ensemble import RandomForestRegressor
# from matplotlib import pyplot

# # transform a time series dataset into a supervised learning dataset
# def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
#         n_vars = 1 if type(data) is list else data.shape[1]
#         df = DataFrame(data)
#         cols = list()
#         # input sequence (t-n, ... t-1)
#         for i in range(n_in, 0, -1):
#                 cols.append(df.shift(i))
#         # forecast sequence (t, t+1, ... t+n)
#         for i in range(0, n_out):
#                 cols.append(df.shift(-i))
#         # put it all together
#         agg = concat(cols, axis=1)
#         # drop rows with NaN values
#         if dropnan:
#                 agg.dropna(inplace=True)
#         return agg.values

# # split a univariate dataset into train/test sets
# def train_test_split(data, n_test):
#         return data[:-n_test, :], data[-n_test:, :]

# # fit an random forest model and make a one step prediction
# def random_forest_forecast(train, testX):
#         # transform list into array
#         train = asarray(train)
#         # split into input and output columns
#         trainX, trainy = train[:, :-1], train[:, -1]
#         # fit model
#         model = RandomForestRegressor(n_estimators=1000)
#         model.fit(trainX, trainy)
#         # make a one-step prediction
#         yhat = model.predict([testX])
#         return yhat[0]

# # walk-forward validation for univariate data
# def walk_forward_validation(data, n_test):
#         predictions = list()
```

```
#         # split dataset
#         train, test = train_test_split(data, n_test)
#         # seed history with training dataset
#         history = [x for x in train]
#         # step over each time-step in the test set
#         for i in range(len(test)):
#                 # split test row into input and output columns
#                 testX, testy = test[i, :-1], test[i, -1]
#                 # fit model on history and make a prediction
#                 yhat = random_forest_forecast(history, testX)
#                 # store forecast in list of predictions
#                 predictions.append(yhat)
#                 # add actual observation to history for the next loop
#                 history.append(test[i])
#                 # summarize progress
#                 print('>expected=%.1f, predicted=%.1f' % (testy, yhat))
#         # estimate prediction error
#         error = mean_absolute_error(test[:, -1], predictions)
#         return error, test[:, -1], predictions

# # load the dataset
# series = pd.read_csv("C:\\Users\\Imran\\Desktop\\yield_df.csv", header=0,␣
 ↪index_col=0)
# values = series.values
# # transform the time series data into supervised learning
# data = series_to_supervised(values, n_in=6)
# # evaluate
# mae, y, yhat = walk_forward_validation(data, 12)
# print('MAE: %.3f' % mae)
# # plot expected vs predicted
# pyplot.plot(y, label='Expected')
# pyplot.plot(yhat, label='Predicted')
# pyplot.legend()
# pyplot.show()
```

2 New Code

[1]:
```
# finalize model and make a prediction for yield with random forest
from numpy import asarray
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.ensemble import RandomForestRegressor

# transform a time series dataset into a supervised learning dataset
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
```

```
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
    agg = pd.concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

[2]:
```
import pandas as pd
series = pd.read_csv("C:\\Users\\Imran\\Desktop\\yield_df.csv", header=0,␣
↪index_col=0)
```

[3]:
```
series.columns
```

[3]:
```
Index(['Area', 'Item', 'Year', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[4]:
```
series
```

[4]:

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	1990	36613	1485.0	121.00	16.37
1	Albania	Potatoes	1990	66667	1485.0	121.00	16.37
2	Albania	Rice, paddy	1990	23333	1485.0	121.00	16.37
3	Albania	Sorghum	1990	12500	1485.0	121.00	16.37
4	Albania	Soybeans	1990	7000	1485.0	121.00	16.37
…	…	…	…	…			
28237	Zimbabwe	Rice, paddy	2013	22581			
28238	Zimbabwe	Sorghum	2013	3066			
28239	Zimbabwe	Soybeans	2013	13142			
28240	Zimbabwe	Sweet potatoes	2013	22222			
28241	Zimbabwe	Wheat	2013	22888			

```
…                              …                 …        …
28237                          657.0             2550.07     19.76
28238                          657.0             2550.07     19.76
28239                          657.0             2550.07     19.76
28240                          657.0             2550.07     19.76
28241                          657.0             2550.07     19.76

[28242 rows x 7 columns]
```

[5]:
```
series.columns
```

[5]:
```
Index(['Area', 'Item', 'Year', 'hg/ha_yield', 'average_rain_fall_mm_per_year',
       'pesticides_tonnes', 'avg_temp'],
      dtype='object')
```

[6]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

series['Area'] = le.fit_transform(series['Area'])
series['Item'] = le.fit_transform(series['Item'])

print(series.tail())
values = series.values
```

```
       Area  Item  Year  hg/ha_yield  average_rain_fall_mm_per_year  \
28237   100     4  2013        22581                          657.0
28238   100     5  2013         3066                          657.0
28239   100     6  2013        13142                          657.0
28240   100     7  2013        22222                          657.0
28241   100     8  2013        22888                          657.0

       pesticides_tonnes  avg_temp
28237            2550.07     19.76
28238            2550.07     19.76
28239            2550.07     19.76
28240            2550.07     19.76
28241            2550.07     19.76
```

[7]:
```
#Scaling Features
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
values=scaler.fit_transform(values)
```

[8]:
```
values
```

[8]:
```
array([[0.00000000e+00, 1.11111111e-01, 0.00000000e+00, …,
        4.49670743e-01, 3.28894097e-04, 5.13458262e-01],
```

```
       [0.00000000e+00, 3.33333333e-01, 0.00000000e+00, …,
        4.49670743e-01, 3.28894097e-04, 5.13458262e-01],
       [0.00000000e+00, 4.44444444e-01, 0.00000000e+00, …,
        4.49670743e-01, 3.28894097e-04, 5.13458262e-01],
       …,
       [1.00000000e+00, 6.66666667e-01, 1.00000000e+00, …,
        1.90028222e-01, 6.93361288e-03, 6.28960818e-01],
       [1.00000000e+00, 7.77777778e-01, 1.00000000e+00, …,
        1.90028222e-01, 6.93361288e-03, 6.28960818e-01],
       [1.00000000e+00, 8.88888889e-01, 1.00000000e+00, …,
        1.90028222e-01, 6.93361288e-03, 6.28960818e-01]])
```

[9]:
```
# transform the time series data into supervised learning
train = series_to_supervised(values, n_in=6)
# split into input and output columns
trainX, trainy = train[:, :-1], train[:, 3]
```

[10]:
```
trainX
```

[10]:
```
array([[0.00000000e+00, 1.11111111e-01, 0.00000000e+00, …,
        5.78783394e-02, 4.49670743e-01, 3.28894097e-04],
       [0.00000000e+00, 3.33333333e-01, 0.00000000e+00, …,
        1.55113471e-01, 4.49670743e-01, 3.28894097e-04],
       [0.00000000e+00, 4.44444444e-01, 0.00000000e+00, …,
        5.68212190e-02, 4.49670743e-01, 3.28894097e-04],
       …,
       [1.00000000e+00, 8.88888889e-01, 9.56521739e-01, …,
        2.61128685e-02, 1.90028222e-01, 6.93361288e-03],
       [1.00000000e+00, 0.00000000e+00, 1.00000000e+00, …,
        4.42235351e-02, 1.90028222e-01, 6.93361288e-03],
       [1.00000000e+00, 1.11111111e-01, 1.00000000e+00, …,
        4.55519166e-02, 1.90028222e-01, 6.93361288e-03]])
```

[11]:
```
trainy
```

[11]:
```
array([0.07292735, 0.13287206, 0.0464395 , …, 0.04860759, 0.09165034,
       0.0147578 ])
```

[12]:
```
# fit model
model = RandomForestRegressor(n_estimators=100)
model.fit(trainX, trainy)
```

[12]:
```
RandomForestRegressor()
```

[13]:
```
import pickle
```

[14]:
```
saved_model = pickle.dump(model, open("C:\\Users\\Imran\\Desktop\\model.pkl",␣
↪"wb"))
```

[15]:
```
#loading save model
rf_saved_model = pickle.load(open("C:\\Users\\Imran\\Desktop\\model.pkl", "rb"))
```

[16]:
```
xtest = series_to_supervised(values, n_in=6)
```

[17]:
```
#testX, testy = xtest[:, 1:], xtest[:, 3]
testX, testy = xtest[:, :-1], xtest[:, 3]
```

[18]:
```
testX
```

[18]:
```
array([[0.00000000e+00, 1.11111111e-01, 0.00000000e+00, …,
        5.78783394e-02, 4.49670743e-01, 3.28894097e-04],
       [0.00000000e+00, 3.33333333e-01, 0.00000000e+00, …,
        1.55113471e-01, 4.49670743e-01, 3.28894097e-04],
       [0.00000000e+00, 4.44444444e-01, 0.00000000e+00, …,
        5.68212190e-02, 4.49670743e-01, 3.28894097e-04],
       …,
       [1.00000000e+00, 8.88888889e-01, 9.56521739e-01, …,
        2.61128685e-02, 1.90028222e-01, 6.93361288e-03],
       [1.00000000e+00, 0.00000000e+00, 1.00000000e+00, …,
        4.42235351e-02, 1.90028222e-01, 6.93361288e-03],
       [1.00000000e+00, 1.11111111e-01, 1.00000000e+00, …,
        4.55519166e-02, 1.90028222e-01, 6.93361288e-03]])
```

[19]:
```
testy
```

[19]:
```
array([0.07292735, 0.13287206, 0.0464395 , …, 0.04860759, 0.09165034,
       0.0147578 ])
```

[20]:
```
y_pred = rf_saved_model.predict(testX)
```

[21]:
```
from sklearn.metrics import mean_absolute_error
```

[22]:
```
mean_absolute_error(testy, y_pred)
```

[22]:
```
1.0953371685360516e-05
```

[23]:
```
from sklearn.metrics import r2_score
score = r2_score(testy, y_pred)
score
```

[23]:
```
0.9999995386548368
```

[37]:
```
row = values[:48].flatten()
# make a one-step prediction
yhat = rf_saved_model.predict(asarray([row]))
print('Input: %s, Predicted: %.3f' % (row, yhat[0]))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-37-6ff6a9e2ceb2> in <module>
      1 row = values[:48].flatten()
      2 # make a one-step prediction
----> 3 yhat = rf_saved_model.predict(asarray([row]))
      4 print('Input: %s, Predicted: %.3f' % (row, yhat[0]))

E:\Git\lib\site-packages\sklearn\ensemble\_forest.py in predict(self, X)
    781         check_is_fitted(self)
    782         # Check data
--> 783         X = self._validate_X_predict(X)
    784
    785         # Assign chunk of trees to jobs

E:\Git\lib\site-packages\sklearn\ensemble\_forest.py in␣
↪_validate_X_predict(self, X)
    419         check_is_fitted(self)
    420
--> 421         return self.estimators_[0]._validate_X_predict(X,␣
↪check_input=True)
    422
    423     @property

E:\Git\lib\site-packages\sklearn\tree\_classes.py in _validate_X_predict(self,␣
↪X, check_input)
    394         n_features = X.shape[1]
    395         if self.n_features_ != n_features:
--> 396             raise ValueError("Number of features of the model must "
    397                              "match the input. Model n_features is %s␣
↪and "
    398                              "input n_features is %s "

ValueError: Number of features of the model must match the input. Model␣
↪n_features is 48 and input n_features is 336
```

[ ]:
```

```