In [16]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [17]:

```
convert_data = pd.read_csv("forced_convert.csv")
```

In [18]:

```
convert_data.head()
```

Out[18]:

	District	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
0	Badin*	109.0	156.0	151.0	195.0	196.0	133.0	109.0	121.0	197.0	211.0	175.0	40.0
1	Tando Allahyar	NaN	26.0	23.0	22.0	20.0	178.0	NaN	13.0	86.0	130.0	73.0	74.0
2	Mirpurkhas	NaN	7.0	NaN	36.0	34.0	49.0	6.0	66.0	101.0	71.0	58.0	67.0
3	Thatta*	40.0	40.0	40.0	40.0	40.0	40.0	40.0	42.0	45.0	41.0	29.0	NaN
4	Umerkot	NaN	NaN	NaN	16.0	NaN	4.0	1.0	49.0	106.0	77.0	51.0	9.0

In [19]:

```
convert_data.tail()
```

Out[19]:

	District	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	202
29	Khanewal	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	Na
30	Lahore	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	Na
31	Multan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1
32	Islamabad	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	Na
33	Sadiqabad	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	Na

In [20]:

```
convert_data.shape
```

Out[20]:

```
(34, 15)
```

In [21]:

```
convert_data.columns
```

Out[21]:

```
Index(['District', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
       '2015', '2016', '2017', '2018', '2019', '2020', 'Total_District_Wis
e'],
      dtype='object')
```

In [22]:

```
convert_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34 entries, 0 to 33
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   District             34 non-null     object
 1   2008                 2 non-null      float64
 2   2009                 4 non-null      float64
 3   2010                 7 non-null      float64
 4   2011                 7 non-null      float64
 5   2012                 9 non-null      float64
 6   2013                 12 non-null     float64
 7   2014                 7 non-null      float64
 8   2015                 16 non-null     float64
 9   2016                 18 non-null     float64
 10  2017                 17 non-null     float64
 11  2018                 22 non-null     float64
 12  2019                 19 non-null     float64
 13  2020                 25 non-null     float64
 14  Total_District_Wise  34 non-null     int64
dtypes: float64(13), int64(1), object(1)
memory usage: 4.1+ KB
```

In [23]:

```
convert_data.describe()
```

Out[23]:

	2008	2009	2010	2011	2012	2013	2014	
count	2.000000	4.00000	7.000000	7.000000	9.000000	12.000000	7.000000	1
mean	74.500000	57.25000	38.857143	45.857143	38.000000	40.083333	22.714286	2
std	48.790368	67.20801	50.814790	67.098009	60.541308	57.892468	40.639530	3
min	40.000000	7.00000	7.000000	6.000000	1.000000	2.000000	1.000000	
25%	57.250000	21.25000	11.000000	11.000000	10.000000	3.000000	1.000000	
50%	74.500000	33.00000	23.000000	22.000000	20.000000	8.000000	1.000000	
75%	91.750000	69.00000	34.500000	38.000000	34.000000	49.500000	23.000000	2
max	109.000000	156.00000	151.000000	195.000000	196.000000	178.000000	109.000000	12

In [24]:

```
convert_data.isnull().sum()
```

Out[24]:

```
District               0
2008                  32
2009                  30
2010                  27
2011                  27
2012                  25
2013                  22
2014                  27
2015                  18
2016                  16
2017                  17
2018                  12
2019                  15
2020                   9
Total_District_Wise    0
dtype: int64
```

In [25]:

```
for i in convert_data.columns:
    convert_data[i] = convert_data[i].fillna(0)
```

In [26]:

```
convert_data.isnull().sum()
```

Out[26]:

```
District              0
2008                  0
2009                  0
2010                  0
2011                  0
2012                  0
2013                  0
2014                  0
2015                  0
2016                  0
2017                  0
2018                  0
2019                  0
2020                  0
Total_District_Wise   0
dtype: int64
```

In [27]:

```
convert_data.head()
```

Out[27]:

	District	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
0	Badin*	109.0	156.0	151.0	195.0	196.0	133.0	109.0	121.0	197.0	211.0	175.0	40.0
1	Tando Allahyar	0.0	26.0	23.0	22.0	20.0	178.0	0.0	13.0	86.0	130.0	73.0	74.0
2	Mirpurkhas	0.0	7.0	0.0	36.0	34.0	49.0	6.0	66.0	101.0	71.0	58.0	67.0
3	Thatta*	40.0	40.0	40.0	40.0	40.0	40.0	40.0	42.0	45.0	41.0	29.0	0.0
4	Umerkot	0.0	0.0	0.0	16.0	0.0	4.0	1.0	49.0	106.0	77.0	51.0	9.0

In [29]:

```
forced_convert_by_district = convert_data.groupby('District').sum()
```

In [30]:

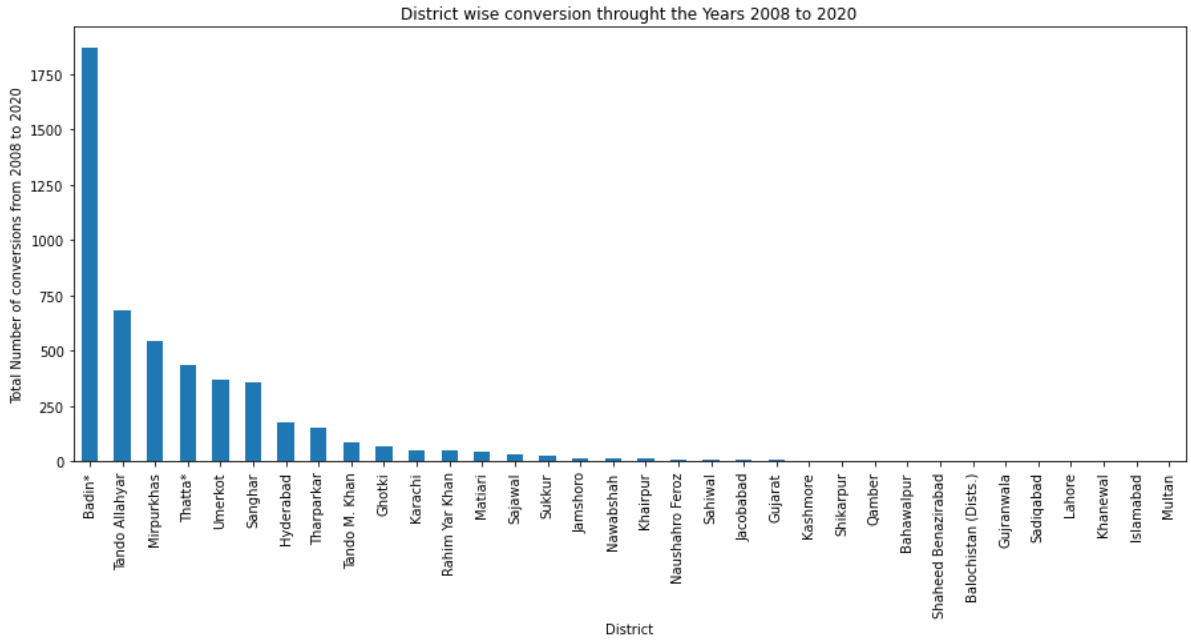
```
forced_convert_by_district.head()
```

Out[30]:

District	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2
Badin*	109.0	156.0	151.0	195.0	196.0	133.0	109.0	121.0	197.0	211.0	175.0	40.0	
Bahawalpur	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	
Balochistan (Dists.)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Ghotki	0.0	0.0	11.0	0.0	0.0	3.0	0.0	1.0	3.0	2.0	6.0	21.0	
Gujarat	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

In [31]:

```
plt.subplots(figsize = (15, 6))
cr = forced_convert_by_district['Total_District_Wise'].sort_values(ascending = False)
ax = cr.plot.bar()
ax.set_xlabel('District')
ax.set_ylabel('Total Number of conversions from 2008 to 2020')
ax.set_title('District wise conversion throught the Years 2008 to 2020')
plt.show()
print(cr)
```




```
District
Badin*                  1870
Tando Allahyar           682
Mirpurkhas               544
Thatta*                  438
Umerkot                  372
Sanghar                  357
Hyderabad                178
Tharparkar               153
Tando M. Khan             88
Ghotki                    69
Karachi                   51
Rahim Yar Khan            48
Matiari                   42
Sajawal                   33
Sukkur                    25
Jamshoro                  17
Nawabshah                 14
Khairpur                  14
Naushahro Feroz           11
Sahiwal                   10
Jacobabad                  9
Gujarat                    7
Kashmore                   5
Shikarpur                  5
Qamber                     2
Bahawalpur                 2
Shaheed Benazirabad        2
```

```
Balochistan (Dists.)       1
Gujranwala                 1
Sadiqabad                  1
Lahore                     1
Khanewal                   1
Islamabad                  1
Multan                     1
Name: Total_District_Wise, dtype: int64
```

In [32]:

```
forced_year_data = pd.read_csv('forced_convert_year.csv')
```

In [33]:

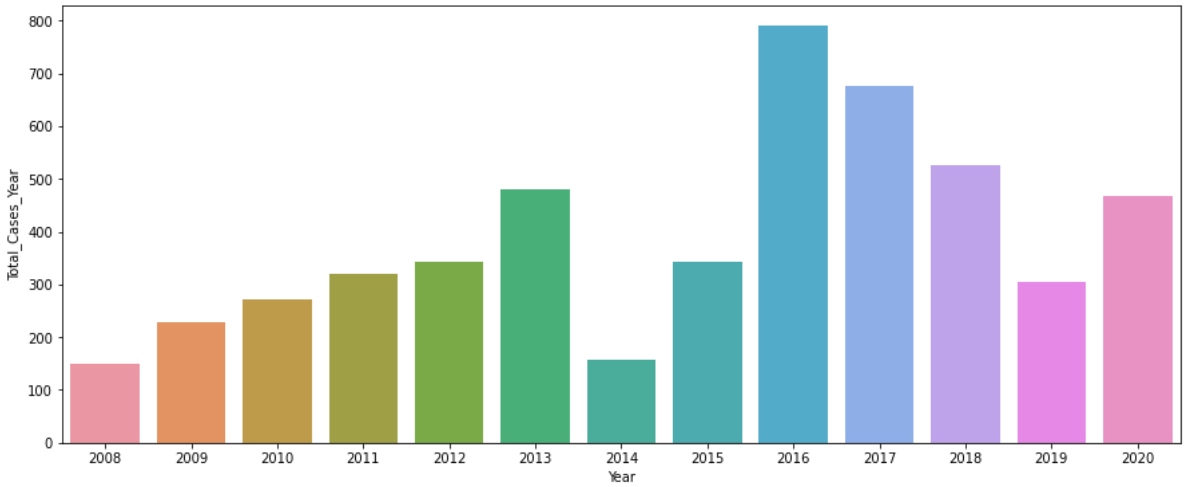
```
forced_year_data
```

Out[33]:

	Year	Total_Cases_Year
0	2008	149
1	2009	229
2	2010	272
3	2011	321
4	2012	342
5	2013	481
6	2014	157
7	2015	344
8	2016	790
9	2017	675
10	2018	525
11	2019	305
12	2020	467

In [36]:

```
plt.figure(figsize=(15,6))
sns.barplot(y = 'Total_Cases_Year', x = 'Year', data = forced_year_data)
plt.xticks(rotation = 0)
plt.show()
```



In [38]:

```
x = convert_data.drop(['District', 'Total_District_Wise'], axis = 1)
y = convert_data['Total_District_Wise']
```

In [39]:

```
x.shape
```

Out[39]:

```
(34, 13)
```

In [40]:

```
y.shape
```

Out[40]:

```
(34,)
```

In [41]:

```
from sklearn.linear_model import LogisticRegression
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)
```

In [42]:

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[42]:

```
LogisticRegression()
```

In [43]:

```
y_pred = model.predict(X_test)
```

In [44]:

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.9629629629629629
Testing Accuracy : 0.14285714285714285
```

In [45]:

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

In [46]:

```
ion report for classifier %s:\n%s\n" % (model,
                                        metrics.classification_report(y_test, y_pred)))
```

```
 Classification report for classifier LogisticRegression():
              precision    recall  f1-score   support

           1       0.33      1.00      0.50         1
           2       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         0
          11       0.00      0.00      0.00         1
          25       0.00      0.00      0.00         0
          33       0.00      0.00      0.00         0
          42       0.00      0.00      0.00         1
         178       0.00      0.00      0.00         1
         438       0.00      0.00      0.00         1
        1870       0.00      0.00      0.00         0

    accuracy                           0.14         7
   macro avg       0.03      0.09      0.05         7
weighted avg       0.05      0.14      0.07         7
```

In [51]:

```
from sklearn.tree import DecisionTreeRegressor
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)
```

In [52]:

```
model1 = DecisionTreeRegressor(max_depth=6)
model1.fit(X_train, y_train)
```

Out[52]:

```
DecisionTreeRegressor(max_depth=6)
```

In [53]:

```
y_pred = model1.predict(X_test)
```

In [54]:

```
print("Training Accuracy :", model1.score(X_train, y_train))
print("Testing Accuracy :", model1.score(X_test, y_test))
```

```
Training Accuracy : 0.9999979322050068
Testing Accuracy : 0.4012366871026326
```

In [56]:

```
from sklearn.naive_bayes import GaussianNB
```

In [57]:

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[57]:

```
GaussianNB()
```

In [58]:

```
y_pred = classifier.predict(X_test)
```

In [59]:

```
print("Training Accuracy :", classifier.score(X_train, y_train))
print("Testing Accuracy :", classifier.score(X_test, y_test))
```

```
Training Accuracy : 0.8888888888888888
Testing Accuracy : 0.2857142857142857
```

In [60]:

```
ion report for classifier %s:\n%s\n" % (classifier,
                                        metrics.classification_report(y_test, y_pred)))
```

```
Classification report for classifier GaussianNB():
              precision    recall  f1-score   support

           1       1.00      0.50      0.67         2
           2       0.50      1.00      0.67         1
           5       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         1
          14       0.00      0.00      0.00         0
          48       0.00      0.00      0.00         1
         682       0.00      0.00      0.00         0
        1870       0.00      0.00      0.00         1

    accuracy                           0.29         7
   macro avg       0.19      0.19      0.17         7
weighted avg       0.36      0.29      0.29         7
```