In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
original=pd.read_csv(r"C:\Users\Benedict Arora\Downloads\crime.csv",engine='pyth
```

In [10]:
```
original['YEAR'].unique()
```

Out[10]:
```
array([2018, 2017, 2016, 2015], dtype=int64)
```

In [11]:
```
original['OCCURRED_ON_DATE']=pd.to_datetime(original['OCCURRED_ON_DATE'])
```

In [12]:
```
np.isnan(original.any())
```

Out[12]:
```
INCIDENT_NUMBER        False
OFFENSE_CODE           False
OFFENSE_CODE_GROUP     False
OFFENSE_DESCRIPTION    False
DISTRICT               False
REPORTING_AREA         False
SHOOTING               False
OCCURRED_ON_DATE       False
YEAR                   False
MONTH                  False
DAY_OF_WEEK            False
HOUR                   False
UCR_PART               False
STREET                 False
Lat                    False
Long                   False
Location               False
dtype: bool
```

In [13]:
```
original['DISTRICT'].unique()
```

Out[13]:
```
array(['E18', 'D14', 'B2', 'A1', 'A7', 'C11', nan, 'D4', 'E13', 'B3',
       'C6', 'A15', 'E5'], dtype=object)
```

Top 5 offense_code_group

In [15]:
```
offense=pd.pivot_table(original.loc[:,['OFFENSE_CODE_GROUP','YEAR','OFFENSE_CODE
                columns='YEAR',aggfunc=np.count_nonzero)
```

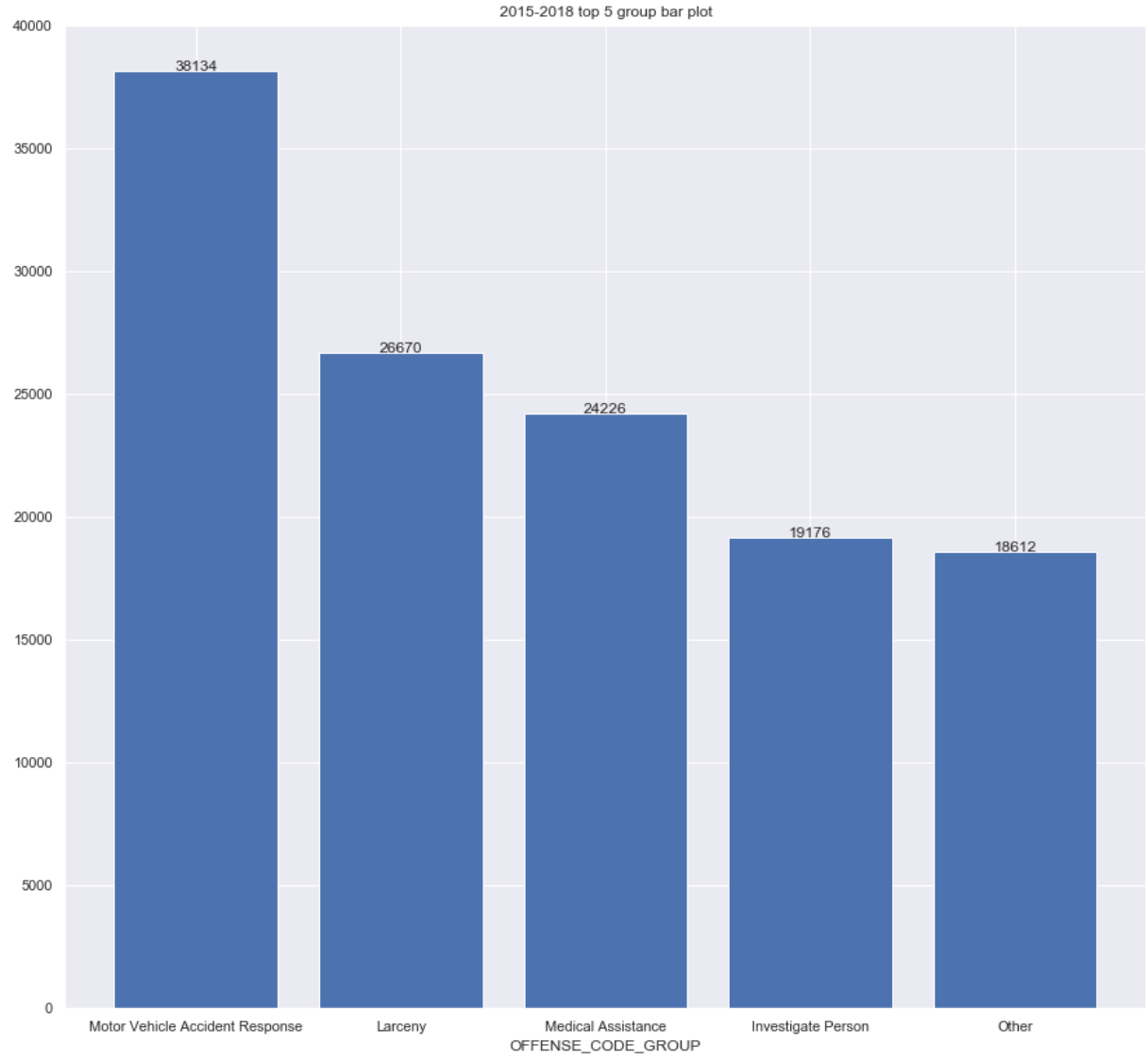
In [16]:
```
summary=pd.DataFrame(offense.apply(np.sum,axis=1))
summary=summary.rename(columns={0:'total'})
```

In [17]:
```
sumsort=summary.sort_values(by='total',ascending=False)
top5=sumsort.iloc[0:5,:]
top5
```

Out[17]:

OFFENSE_CODE_GROUP	total
Motor Vehicle Accident Response	38134.0
Larceny	26670.0
Medical Assistance	24226.0
Investigate Person	19176.0
Other	18612.0

In [18]:
```
sns.set()
p0=plt.figure(figsize=(15,14))
plt.title(r'2015-2018 top 5 group bar plot')
plt.bar(range(top5.index.shape[0]),top5.loc[:,'total'])
plt.xlabel('OFFENSE_CODE_GROUP')
plt.xticks(range(top5.index.shape[0]),top5.index)
x=np.arange(top5.index.shape[0])
y=np.array(top5['total'])
for i,j in zip(x,y):
    plt.text(i,j,'%d'%j,ha='center')
p0.savefig('./top5_group_bar.png')
plt.show()
```



visualization by heading

In [19]:
```
disgroup=original.groupby(by='DISTRICT')
```

In [20]:
```
groupcount=disgroup.count()
```

In [21]:
```
groupcount.head()
```

Out[21]:

DISTRICT	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION
A1	36735	36735	36735	36735
A15	6663	6663	6663	6663
A7	13634	13634	13634	13634
B2	51288	51288	51288	51288
B3	36400	36400	36400	36400

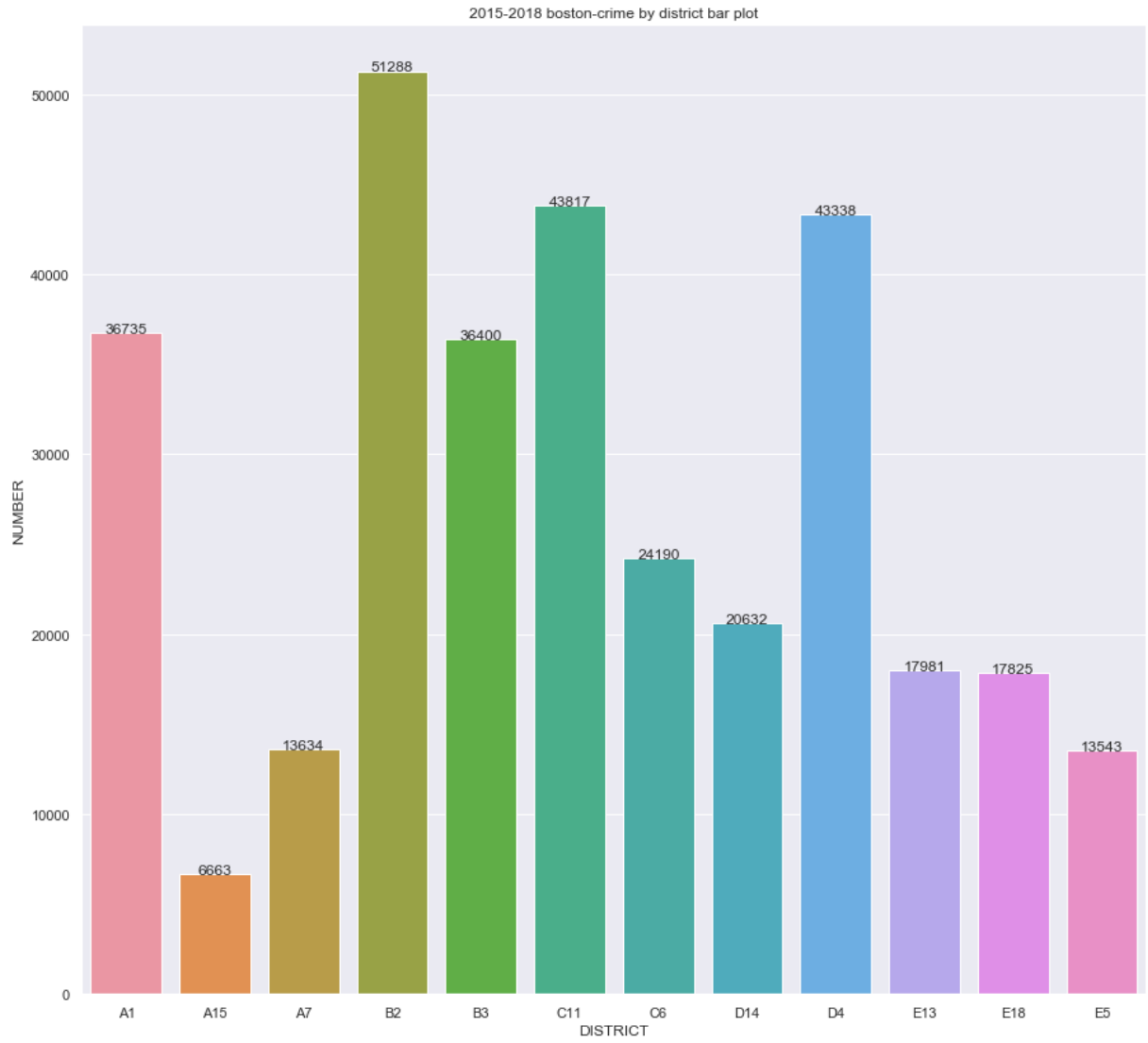
In [23]:
```
number=groupcount.iloc[:,0]
number=pd.DataFrame(number)
```

In [24]:
```
number.rename(columns={'INCIDENT_NUMBER':'NUMBER'},inplace=True)
number.head()
```

Out[24]:

DISTRICT	NUMBER
A1	36735
A15	6663
A7	13634
B2	51288
B3	36400

In [25]:
```
plt.figure(figsize=(15,14))
plt.title(r'2015-2018 boston-crime by district bar plot')
p1=sns.barplot(x=number.index,y='NUMBER',data=number)
x=np.arange(number.index.shape[0])
y=np.array(list(number['NUMBER']))
for i,j in zip(x,y):
    plt.text(i,j+0.05,'%d'%j,ha='center')
else:
    pass
p1fig=p1.get_figure()
p1fig.savefig('./total_by_district_bar.png')
plt.show()
```



DISTRICT B2 district has highest number of crime

C11 and D4 are higher than other district

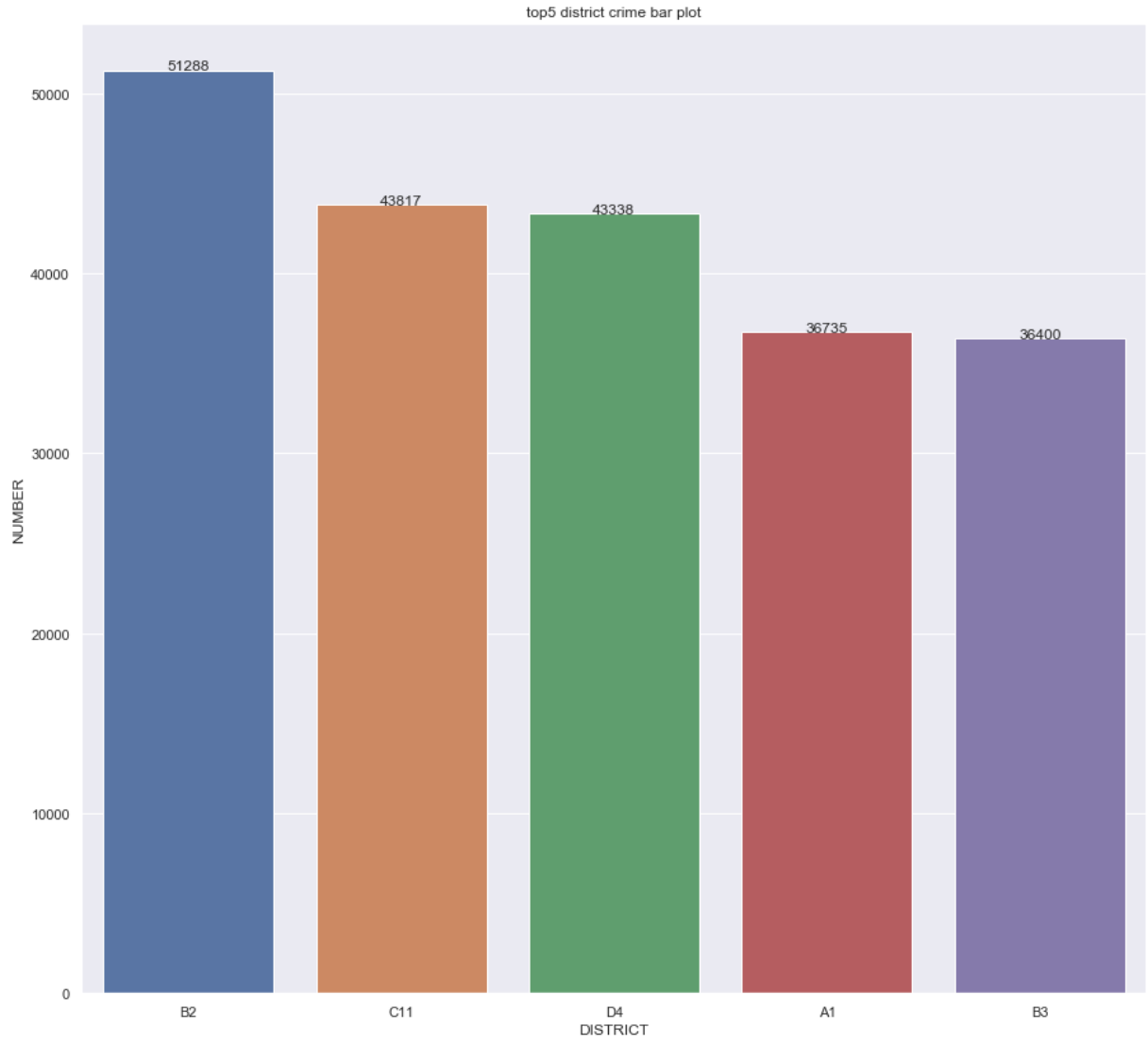
A15 is least

follow this,from number dataframe take top5 district

In [26]:
```
districtsorted=number.sort_values(by='NUMBER',ascending=False)
```

In [27]:
```
top5=districtsorted.iloc[0:5,:]
```

In [28]:
```
plt.figure(figsize=(15,14))
plt.title(r'top5 district crime bar plot')
p2=sns.barplot(x=top5.index,y='NUMBER',data=top5)
x=np.arange(top5.index.shape[0])
y=np.array(list(top5['NUMBER']))
for i,j in zip(x,y):
    plt.text(i,j+0.05,'%d'%j,ha='center')
else:
    pass
p2fig=p2.get_figure()
p2fig.savefig('./top5_district_crime_bar.png')
plt.show()
```



visualization by year

In [29]:
```
yeargroup=original.groupby(by='YEAR')
```

In [30]:
```
count=yeargroup.count()
```

In [31]:
```
yearnumber=pd.DataFrame(count.iloc[:,0])
```

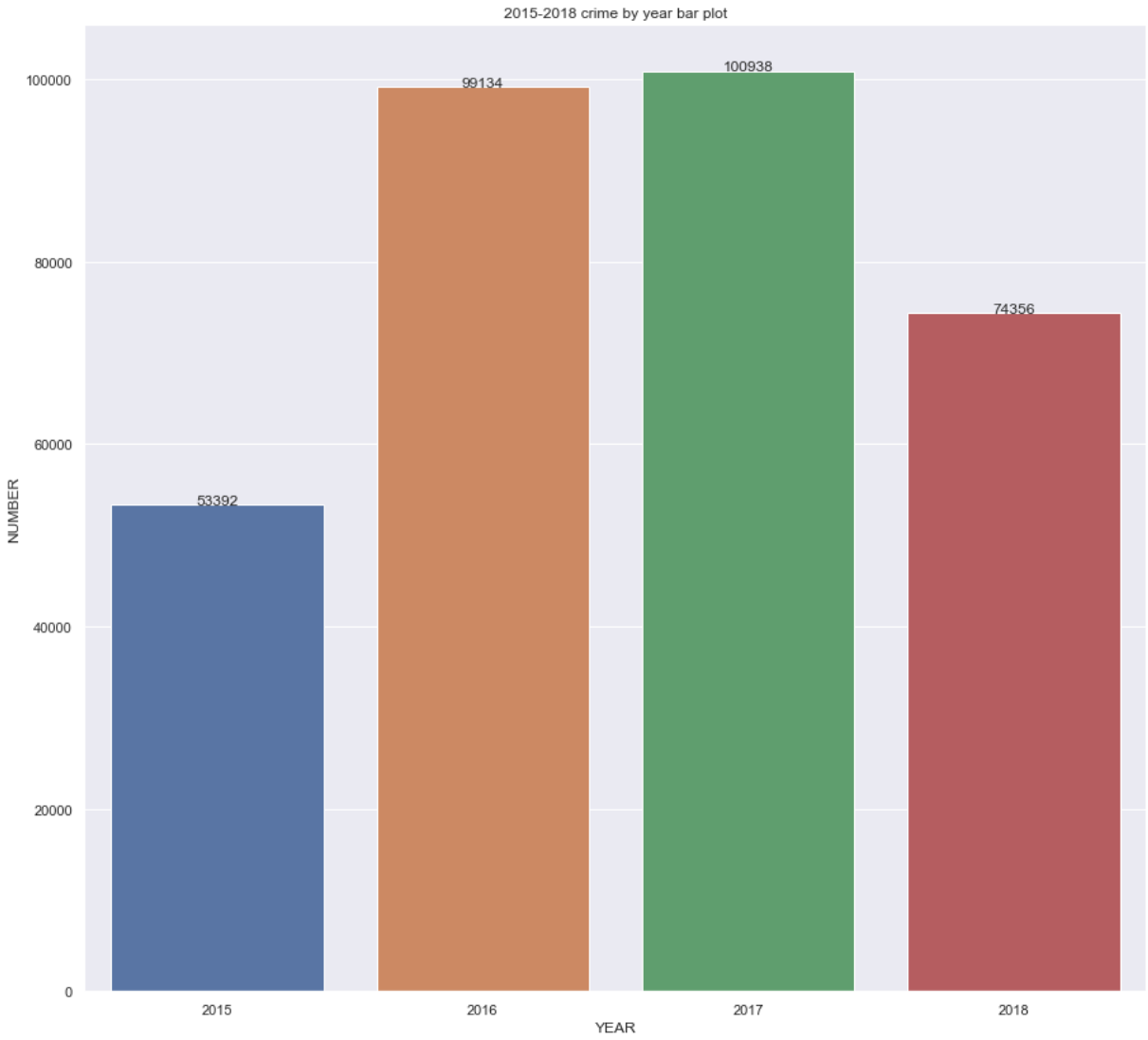
In [32]:
```
yearnumber.rename(columns={'INCIDENT_NUMBER':'NUMBER'},inplace=True)
```

In [33]:
```
yearnumber
```

Out[33]:

YEAR	NUMBER
2015	53392
2016	99134
2017	100938
2018	74356

In [34]:
```
plt.figure(figsize=(15,14))
plt.title(r'2015-2018 crime by year bar plot')
p3=sns.barplot(x=yearnumber.index,y='NUMBER',data=yearnumber)
x=np.arange(yearnumber.index.shape[0])
y=np.array(list(yearnumber['NUMBER']))
for i,j in zip(x,y):
    plt.text(i,j,'%d'%j,ha='center')
else:
    pass
p3fig=p3.get_figure()
p3fig.savefig('./total_by_year_bar.png')
plt.show()
```



by year With this bar chart,we can see 2017's crime number is highest

And 2015 is lowest,this may be caused by people get more depressive by year.

As news,we can see more and more crime has happened

This may cause this chart that appears higher trend.

shooting crime summary and visualization

In [35]:
```
original['SHOOTING'].unique()
```

Out[35]:
```
array([nan, 'Y'], dtype=object)
```

In [36]:
```
original.shape
```

Out[36]:
```
(327820, 17)
```

In [37]:
```
original['SHOOTING']=original['SHOOTING'].fillna('N')
```

In [38]:
```
original['SHOOTING'].unique()
```

Out[38]:
```
array(['N', 'Y'], dtype=object)
```

In [39]:
```
original.head()
```

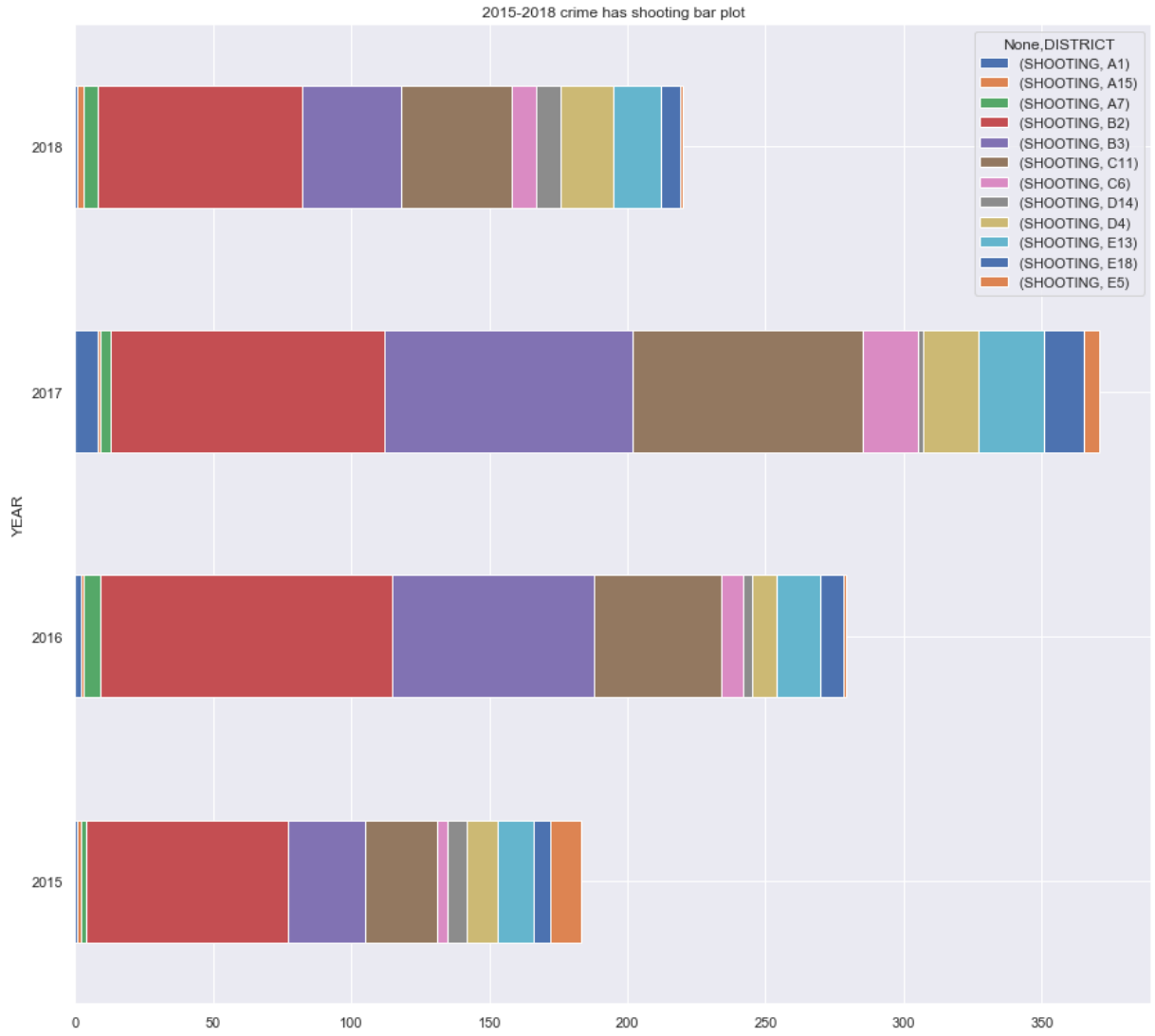
Out[39]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTR
0	I182080058	2403	Disorderly Conduct	DISTURBING THE PEACE	
1	I182080053	3201	Property Lost	PROPERTY - LOST	I
2	I182080052	2647	Other	THREATS TO DO BODILY HARM	
3	I182080051	413	Aggravated Assault	ASSAULT - AGGRAVATED - BATTERY	
4	I182080050	3122	Aircraft	AIRCRAFT INCIDENTS	

In [40]:
```
shootcrime=pd.pivot_table(original.loc[original['SHOOTING']=='Y',['YEAR','DISTRI(
                index='YEAR',columns='DISTRICT',aggfunc=np.count_nonzero)
```

In [41]:
```
sns.set()
p4=shootcrime.plot(title=r'2015-2018 crime has shooting bar plot',figsize=(15,14

p4fig=p4.get_figure()
p4fig.savefig('./total_shooting_crime_barh.png')
plt.show()
```



In [42]:
```
districtSum=shootcrime.apply(np.sum)
districtSum=pd.DataFrame(districtSum)
```

In [43]:
```
districtSum=districtSum.rename(columns={0:r'shooting total'})
```

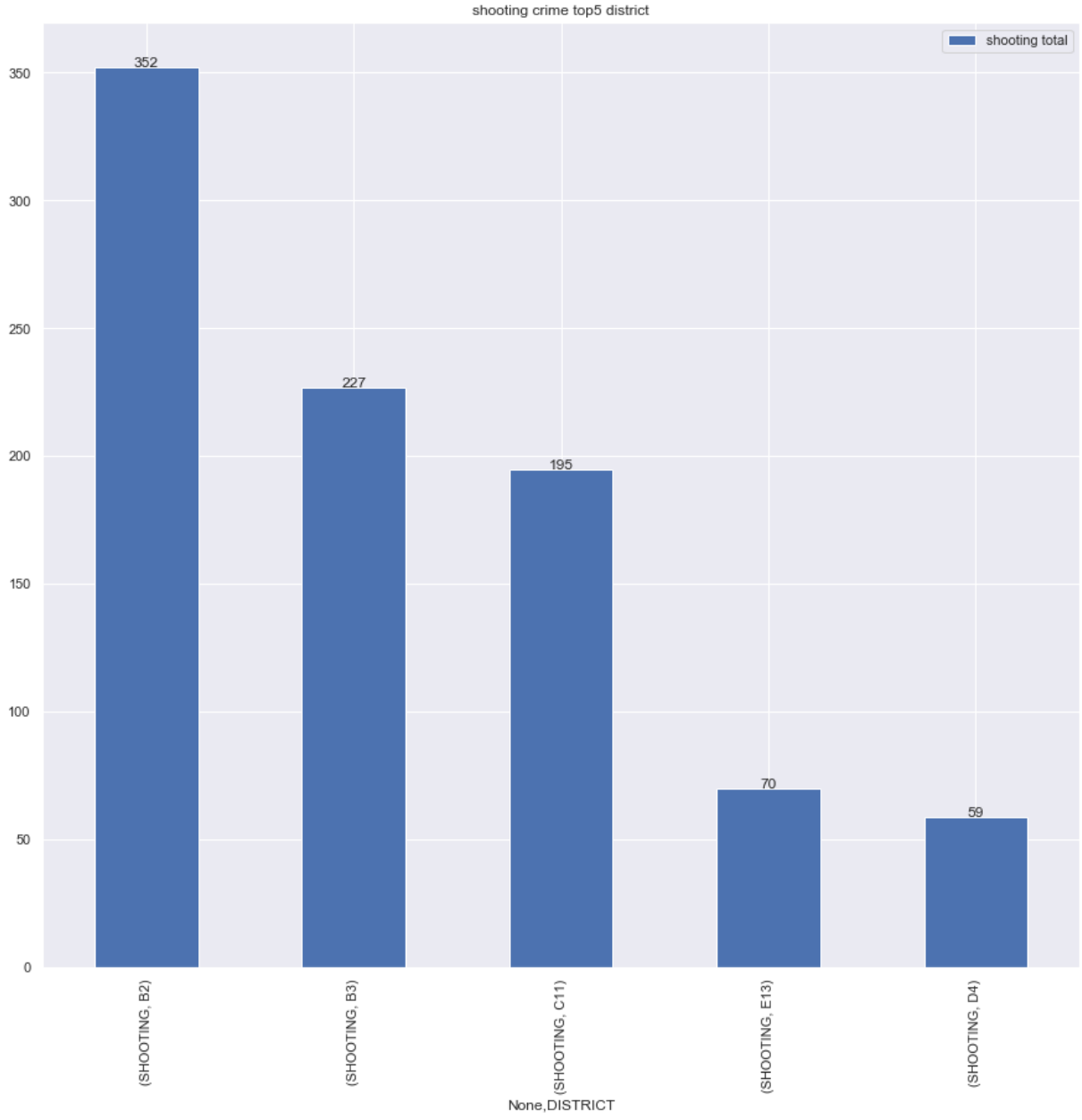
In [44]:
```
districtSum=districtSum.sort_values(by=r'shooting total',ascending=False)
```

In [45]:
```
top5=districtSum.iloc[0:5,:]
top5
```

Out[45]:

		shooting total
	DISTRICT	
SHOOTING	**B2**	352
	B3	227
	C11	195
	E13	70
	D4	59

In [46]:
```
sns.set()
p5=top5.plot(title=r'shooting crime top5 district',figsize=(15,14),kind='bar')
x=np.arange(top5.index.shape[0])
y=np.array(list(top5[r'shooting total']))
for i,j in zip(x,y):
    plt.text(i,j,'%d'%j,ha='center')
p5fig=p5.get_figure()
p5fig.savefig('./shooting_crime_top5_bar.png')
plt.show()
```



shooting crime As this chart,B2 is the highest

and B2's crime number is the highest

We may need to be alert with this distrcit

B3'crime is fifth,but shooting crime is second

With two charts,the more crime happened,the more shooting crime happened.

visualization by month

In [47]:
```
byMonth=original.groupby(by='MONTH')
```

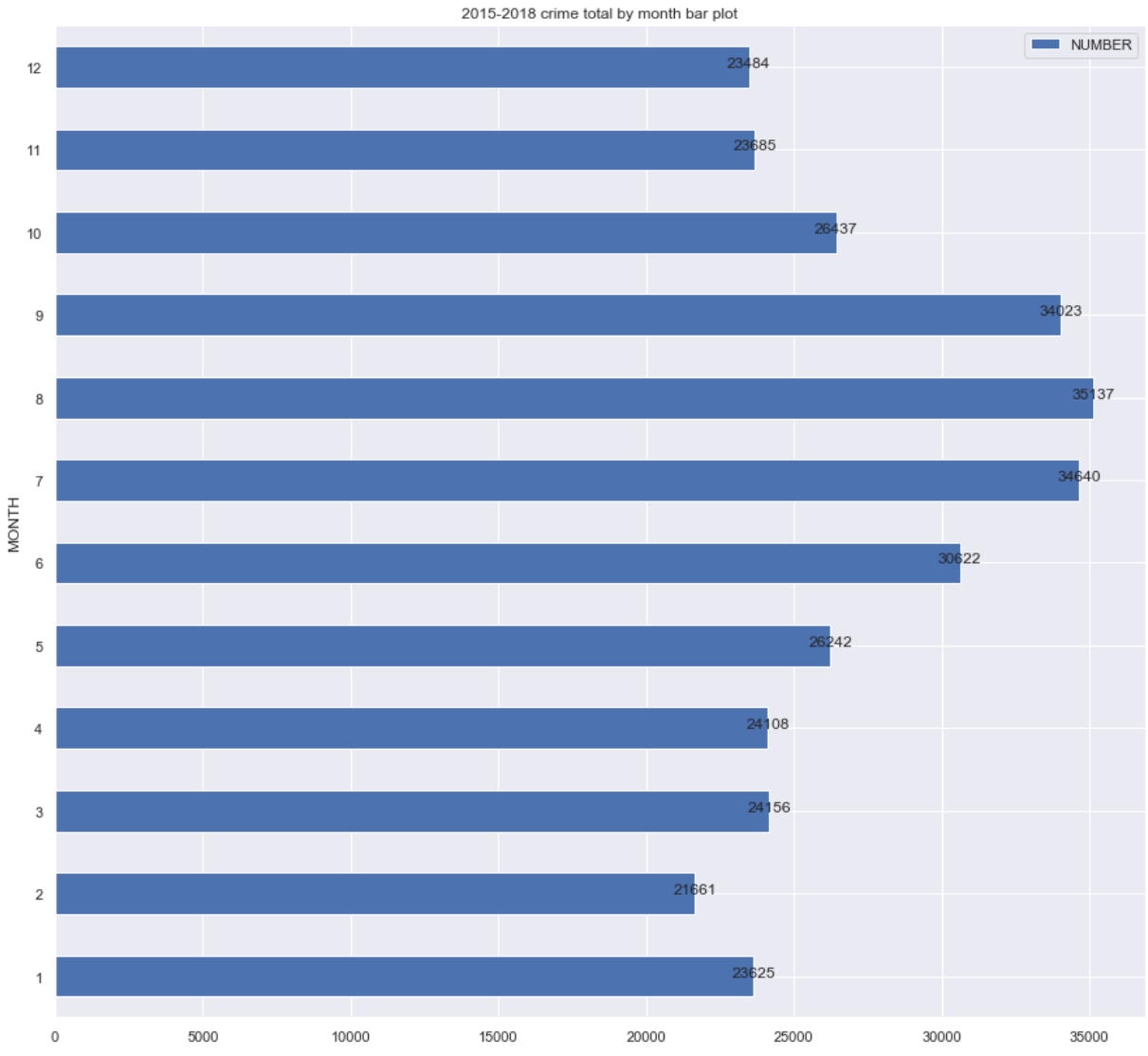
In [48]:
```
Monthcount=byMonth.count()
```

In [49]:
```
MonthNumber=pd.DataFrame(Monthcount.iloc[:,0])
MonthNumber.head()
```

Out[49]:

MONTH	INCIDENT_NUMBER
1	23625
2	21661
3	24156
4	24108
5	26242

In [50]:
```
MonthNumber=MonthNumber.rename(columns={'INCIDENT_NUMBER':'NUMBER'})
sns.set()
p6=MonthNumber.plot(title=r'2015-2018 crime total by month bar plot',figsize=(15
x=np.arange(MonthNumber.index.shape[0])
y=np.array(list(MonthNumber['NUMBER']))
for i,j in zip(x,y):
    plt.text(j,i,'%d'%j,ha='center')
p6fig=p6.get_figure()
p6fig.savefig('./total_by_month_bar.png')
plt.show()
```



by Month With this chart,July~August more crime happened

December~Febuary is less

Should draw a boxplot to see

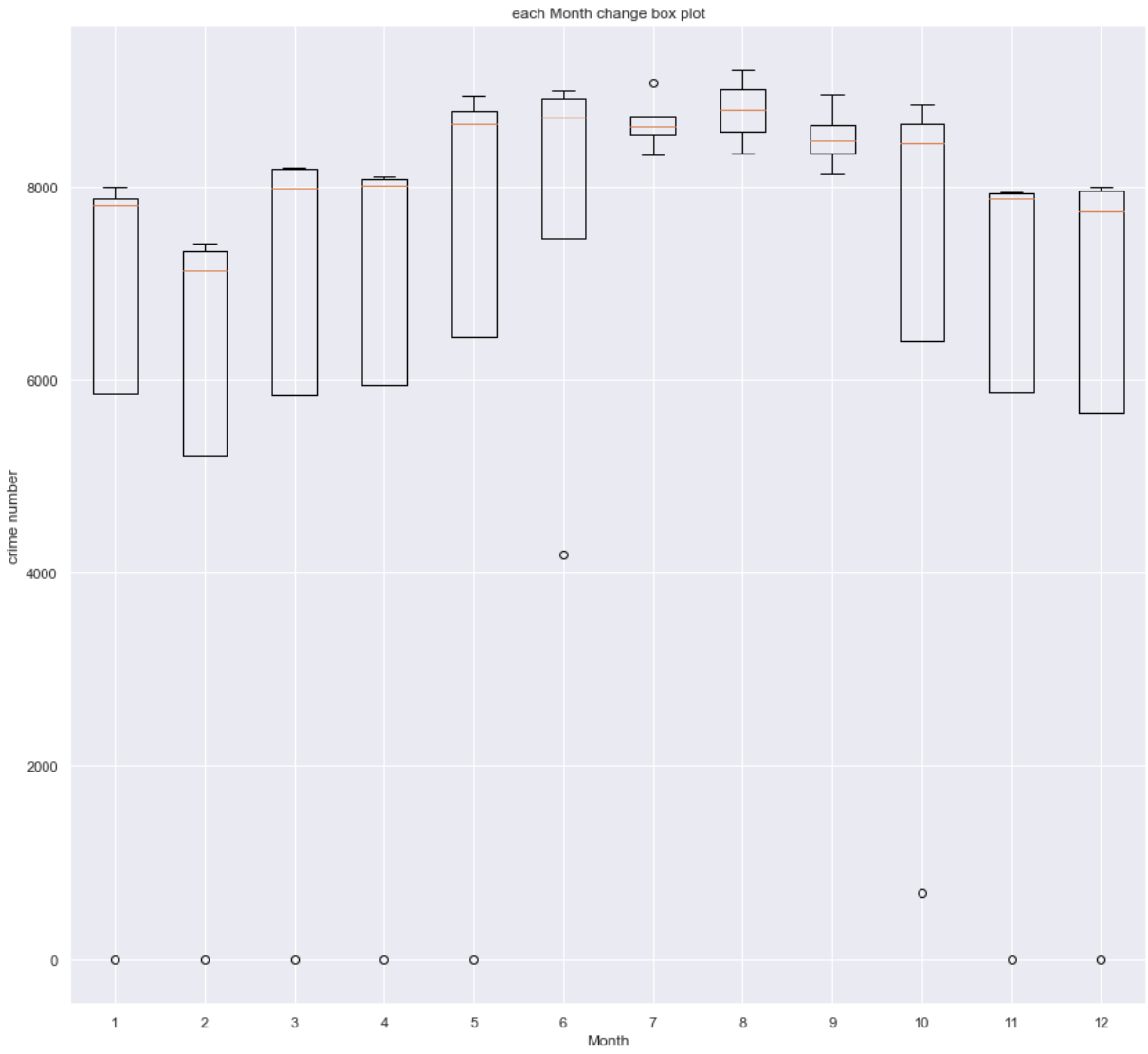
In [51]:
```
Month=pd.pivot_table(original.loc[:,['YEAR','MONTH','INCIDENT_NUMBER']], \
                     index='YEAR',columns='MONTH',aggfunc=np.count_nonzero)
Month
```

Out[51]:

	INCIDENT_NUMBER											
MONTH	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
YEAR												
2015	NaN	NaN	NaN	NaN	NaN	4191.0	8326.0	8343.0	8415.0	8308.0	7818.0	799
2016	7837.0	7310.0	8199.0	8101.0	8582.0	8558.0	8620.0	8940.0	8526.0	8586.0	7924.0	795
2017	7993.0	7408.0	8179.0	8072.0	8721.0	8990.0	9077.0	9209.0	8950.0	8854.0	7943.0	754
2018	7795.0	6943.0	7778.0	7935.0	8939.0	8883.0	8617.0	8645.0	8132.0	689.0	NaN	N

In [52]:
```
Month=Month.fillna(0)
```

In [53]:
```
Monthlist=(list(Month.iloc[:,0])),(list(Month.iloc[:,1])),(list(Month.iloc[:,2]))
(list(Month.iloc[:,3])),(list(Month.iloc[:,4])),(list(Month.iloc[:,5])),(list(Mo
(list(Month.iloc[:,7])),(list(Month.iloc[:,8])),(list(Month.iloc[:,9])),(list(Mo
(list(Month.iloc[:,11]))
def takesecond(elem):
    x=[]
    for i in elem:
        x.append(i[1])
    else:
        return x
label=takesecond(Month.columns)
sns.set()
p1=plt.figure(figsize=(15,14))
plt.boxplot(Monthlist,labels=label,meanline=True)
plt.title(r'each Month change box plot')
plt.xlabel('Month')
plt.ylabel(r'crime number')
p1.savefig('./by_month_boxplot.png')
plt.show()
```



boxplot-Month 2015 January~May and 2018 November~December values are NaN,which is No

record

2018 October is less,which is seen as abnormal value This may be caused by No completed recording

July~Auguest has less change

With month bar chart,they may be crime's higher occurred months

And July in 2017 has high value,which is seen as abnormal value

Other Month have gentle change

Principal component analysis find the most relative features

using PCA model to analysis

In [54]:
```
from sklearn.preprocessing import LabelEncoder
tras=original.iloc[:,:]
tras.loc[:,'OFFENSE_CODE_GROUP']=LabelEncoder().fit_transform(tras.loc[:,'OFFENSI
tras.loc[:,'OFFENSE_DESCRIPTION']=LabelEncoder().fit_transform(tras.loc[:,'OFFENS
tras.loc[:,'DISTRICT']=LabelEncoder().fit_transform(tras.loc[:,'DISTRICT'].astype
tras.loc[:,'SHOOTING']=LabelEncoder().fit_transform(tras.loc[:,'SHOOTING'].astype
tras.loc[:,'DAY_OF_WEEK']=LabelEncoder().fit_transform(tras.loc[:,'DAY_OF_WEEK'])
tras.loc[:,'UCR_PART']=LabelEncoder().fit_transform(tras.loc[:,'UCR_PART'].astype
tras.loc[:,'STREET']=LabelEncoder().fit_transform(tras.loc[:,'STREET'].astype('st
```

In [56]:
```
tras.head()
```

Out[56]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTR
0	I182080058	2403	14	62	
1	I182080053	3201	52	186	
2	I182080052	2647	46	221	
3	I182080051	413	0	16	
4	I182080050	3122	1	4	

In [57]:
```
tras.loc[:,'REPORTING_AREA']=LabelEncoder().fit_transform(tras.loc[:,'REPORTING_A
```

In [58]:
```
data=tras.loc[:,['OFFENSE_CODE', 'OFFENSE_CODE_GROUP','OFFENSE_DESCRIPTION', \
                 'DISTRICT', 'REPORTING_AREA', 'SHOOTING', 'YEAR','DAY_OF_WEEK',
target=tras.loc[:,'MONTH']
```

In [59]:
```
from sklearn.decomposition import PCA
pcamodel=PCA(n_components=11).fit(data)
```

In [60]:
```
print(pcamodel.explained_variance_ratio_)
'''
top2
OFFENSE_CODE
OFFENSE_CODE_GROUP
'''
```

```
[5.91420598e-01 3.89405264e-01 1.81125137e-02 9.92571142e-04
 5.39400301e-05 1.09663313e-05 2.59277550e-06 1.14421814e-06
 2.81928127e-07 1.27758734e-07 8.81706940e-10]
```

Out[60]:
```
'\ntop2\nOFFENSE_CODE\nOFFENSE_CODE_GROUP\n'
```

PCA With using PCA model,OFFENSE_CODE and OFFENSE_CODE_GROUP are main

2019-02-20 Use pearson coeffient to analysis correlation

to choose features to predict

corelation analysis-pearson coefficient

In [61]:
```
pdata=tras.loc[:,['OFFENSE_CODE', 'OFFENSE_CODE_GROUP','OFFENSE_DESCRIPTION', \
                  'DISTRICT', 'REPORTING_AREA', 'SHOOTING', 'YEAR','DAY_OF_WEEK',
pearsonMatrix=pd.DataFrame(round(pdata.corr(method='pearson'),6))
pearsonMatrix.sort_values(by='MONTH',ascending=False)
```

Out[61]:

	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DIS
MONTH	-0.013767	-0.005614	-0.009981	-0.0
SHOOTING	-0.058043	-0.076569	-0.051014	-0.0
DAY_OF_WEEK	-0.002096	-0.001146	0.004687	0.0
HOUR	-0.017109	-0.022071	-0.016076	0.0
REPORTING_AREA	0.015461	0.011612	0.002161	0.1
DISTRICT	0.015447	-0.002160	0.003713	1.0
STREET	0.002873	-0.005105	-0.003713	-0.0
OFFENSE_CODE_GROUP	0.251910	1.000000	0.563958	-0.0
OFFENSE_DESCRIPTION	0.458922	0.563958	1.000000	0.0
OFFENSE_CODE	1.000000	0.251910	0.458922	0.0
UCR_PART	0.226169	0.191804	0.076092	-0.0
YEAR	0.043738	0.012072	0.017729	0.0

pearson coefficient matrix With using pearson coefficient matrix,

SHOOTING,DAT_OF_WEEK and HOUR are more relative to predict Month

try to use three features to predict

predict Month-using GBC using GradientBoostingClassifier

In [62]:
```
sample=tras.sample(n=10000)
```

In [63]:
```
features=sample.loc[:,['SHOOTING','DAY_OF_WEEK','HOUR']]
target=sample.loc[:,'MONTH']
```

In [64]:
```
from sklearn.model_selection import train_test_split
dataTrain,dataTest, \
targetTrain,targetTest= \
train_test_split(features,target,train_size=0.8)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2179: FutureWarning: From version 0.21, test_size will always complement train_size unless both are specified.
  FutureWarning)
```

In [65]:
```
from sklearn.ensemble import GradientBoostingClassifier as GBC
```

In [66]:
```
crimeGBC=GBC(max_depth=12)
```

In [69]:
```
crimeGBC.fit(dataTrain,targetTrain)
```

Out[69]:
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=12,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

In [70]:
```
crimeGBC.fit(dataTrain,targetTrain)
```

Out[70]:
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=12,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

In [71]:
```
pred=crimeGBC.predict(dataTrain)
```

In [72]:
```
from sklearn.metrics import classification_report
print(classification_report(targetTrain,pred))
```

```
              precision    recall  f1-score   support

           1       0.17      0.10      0.13       591
           2       0.15      0.04      0.07       506
           3       0.17      0.06      0.09       553
           4       0.17      0.05      0.08       597
           5       0.18      0.13      0.15       636
           6       0.17      0.24      0.20       778
           7       0.17      0.34      0.23       877
           8       0.16      0.28      0.21       836
           9       0.17      0.23      0.19       789
          10       0.17      0.12      0.14       645
          11       0.14      0.13      0.14       615
          12       0.16      0.07      0.10       577

   micro avg       0.17      0.17      0.17      8000
   macro avg       0.17      0.15      0.14      8000
weighted avg       0.17      0.17      0.15      8000
```

In [73]:
```
predict=crimeGBC.predict(dataTest)
```

In [74]:
```
print(classification_report(targetTest,predict))
```

```
              precision    recall  f1-score   support

           1       0.08      0.04      0.06       139
           2       0.19      0.03      0.05       143
           3       0.14      0.05      0.07       144
           4       0.07      0.02      0.03       165
           5       0.09      0.06      0.07       161
           6       0.08      0.11      0.09       189
           7       0.10      0.23      0.14       207
           8       0.13      0.22      0.16       213
           9       0.08      0.10      0.09       214
          10       0.04      0.05      0.05       123
          11       0.09      0.10      0.09       135
          12       0.09      0.04      0.05       167

   micro avg       0.10      0.10      0.10      2000
   macro avg       0.10      0.09      0.08      2000
weighted avg       0.10      0.10      0.08      2000
```

In [75]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(targetTest,predict))
```

```
0.096
```

predict month using GBC and PCA

In [78]:
```
pcadata=sample.loc[:,['OFFENSE_CODE', 'OFFENSE_CODE_GROUP','OFFENSE_DESCRIPTION',
                      'DISTRICT', 'REPORTING_AREA', 'SHOOTING', 'YEAR','DAY_OF_WEEK',
target=sample.loc[:,'MONTH']
```

In [77]:
```
Pcatras=PCA(n_components=2).fit_transform(pcadata)
```

In [79]:
```
pcaTrain,pcaTest, \
ptargetTrain,ptargetTest = \
train_test_split(Pcatras,target,train_size=0.8)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2179: FutureWarning: From version 0.21, test_size will always complement train_size unless both are specified.
  FutureWarning)
```

In [80]:
```
pcaGBC=GBC(max_depth=12).fit(pcaTrain,ptargetTrain)
```

In [81]:
```
pcapre=pcaGBC.predict(pcaTrain)
print(classification_report(ptargetTrain,pcapre))
```

```
              precision    recall  f1-score   support

           1       1.00      0.99      0.99       608
           2       0.99      1.00      0.99       525
           3       1.00      0.99      0.99       551
           4       0.99      0.99      0.99       603
           5       1.00      1.00      1.00       632
           6       0.99      0.99      0.99       776
           7       1.00      0.99      0.99       840
           8       0.99      0.99      0.99       849
           9       0.99      0.99      0.99       799
          10       0.98      1.00      0.99       636
          11       0.99      0.99      0.99       582
          12       0.99      0.99      0.99       599

   micro avg       0.99      0.99      0.99      8000
   macro avg       0.99      0.99      0.99      8000
weighted avg       0.99      0.99      0.99      8000
```

In [87]:
```
pcapredict=pcaGBC.predict(pcaTest)
print(classification_report(ptargetTest,pcapredict))
print(accuracy_score(ptargetTest,pcapredict))
```

```
              precision    recall  f1-score   support

           1       0.07      0.07      0.07       122
           2       0.10      0.08      0.09       124
           3       0.06      0.05      0.06       146
           4       0.07      0.06      0.07       159
           5       0.08      0.08      0.08       165
           6       0.11      0.11      0.11       191
           7       0.16      0.16      0.16       244
           8       0.11      0.15      0.13       200
           9       0.13      0.14      0.13       204
          10       0.09      0.11      0.09       132
          11       0.08      0.05      0.07       168
          12       0.06      0.06      0.06       145

   micro avg       0.10      0.10      0.10      2000
   macro avg       0.09      0.09      0.09      2000
weighted avg       0.10      0.10      0.10      2000

0.0995
```

In [ ]:
```
```

predict month using svc and three feature

In [83]:
```
from sklearn.svm import SVC
crimeSVC=SVC()
```

In [84]:
```
crimeSVC.fit(dataTrain,targetTrain)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: Fu
tureWarning: The default value of gamma will change from 'auto' to 'scale' in v
ersion 0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[84]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [85]:
```
SVCpre=crimeSVC.predict(dataTrain)
print(classification_report(targetTrain,SVCpre))
```

```
              precision    recall  f1-score   support

           1       0.13      0.01      0.02       591
           2       0.00      0.00      0.00       506
           3       0.24      0.01      0.02       553
           4       0.23      0.02      0.03       597
           5       0.20      0.03      0.05       636
           6       0.14      0.19      0.16       778
           7       0.14      0.52      0.22       877
           8       0.14      0.30      0.19       836
           9       0.14      0.23      0.17       789
          10       0.15      0.04      0.07       645
          11       0.00      0.00      0.00       615
          12       0.00      0.00      0.00       577

   micro avg       0.14      0.14      0.14      8000
   macro avg       0.13      0.11      0.08      8000
weighted avg       0.13      0.14      0.09      8000


C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

predict month using pvc and svc

In [88]:
```
pSVC=SVC().fit(pcaTrain,ptargetTrain)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: Fu
tureWarning: The default value of gamma will change from 'auto' to 'scale' in v
ersion 0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [89]:
```
pcaSVCpre=pSVC.predict(pcaTrain)
print(classification_report(ptargetTrain,pcaSVCpre))
```

```
              precision    recall  f1-score   support

           1       0.92      0.79      0.85       608
           2       0.88      0.71      0.78       525
           3       0.90      0.74      0.81       551
           4       0.89      0.81      0.85       603
           5       0.87      0.84      0.85       632
           6       0.84      0.85      0.84       776
           7       0.71      0.94      0.81       840
           8       0.76      0.92      0.83       849
           9       0.78      0.89      0.83       799
          10       0.85      0.81      0.83       636
          11       0.87      0.74      0.80       582
          12       0.91      0.77      0.83       599

   micro avg       0.83      0.83      0.83      8000
   macro avg       0.85      0.82      0.83      8000
weighted avg       0.84      0.83      0.83      8000
```

In [90]:
```
pcaSVCpredict=pSVC.predict(pcaTest)
print(classification_report(ptargetTest,pcaSVCpredict))
print(accuracy_score(ptargetTest,pcaSVCpredict))
```

```
              precision    recall  f1-score   support

           1       0.05      0.02      0.02       122
           2       0.19      0.06      0.10       124
           3       0.07      0.02      0.03       146
           4       0.06      0.02      0.03       159
           5       0.10      0.04      0.05       165
           6       0.10      0.04      0.06       191
           7       0.13      0.66      0.22       244
           8       0.10      0.08      0.09       200
           9       0.08      0.04      0.06       204
          10       0.08      0.05      0.06       132
          11       0.04      0.01      0.02       168
          12       0.09      0.03      0.04       145

   micro avg       0.11      0.11      0.11      2000
   macro avg       0.09      0.09      0.06      2000
weighted avg       0.09      0.11      0.07      2000

0.1135
```

predict month using Randomforestclassifier and pca

In [91]:
```
from sklearn.ensemble import RandomForestClassifier as RFC
crimeRFC=RFC(max_depth=12)
```

In [92]:
```
crimeRFC
```

Out[92]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=12, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [93]:
```
crimeRFC.fit(pcaTrain,ptargetTrain)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:
246: FutureWarning: The default value of n_estimators will change from 10 in ve
rsion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[93]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=12, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [94]:
```
RFCpre=crimeRFC.predict(pcaTrain)
print(classification_report(ptargetTrain,RFCpre))
```

```
              precision    recall  f1-score   support

           1       0.62      0.35      0.45       608
           2       0.63      0.31      0.41       525
           3       0.67      0.25      0.37       551
           4       0.52      0.28      0.36       603
           5       0.53      0.29      0.37       632
           6       0.44      0.38      0.41       776
           7       0.25      0.54      0.34       840
           8       0.22      0.64      0.33       849
           9       0.45      0.42      0.43       799
          10       0.58      0.23      0.33       636
          11       0.62      0.28      0.38       582
          12       0.59      0.29      0.39       599

   micro avg       0.37      0.37      0.37      8000
   macro avg       0.51      0.35      0.38      8000
weighted avg       0.49      0.37      0.38      8000
```

In [95]:
```
RFCpredict=crimeRFC.predict(pcaTest)
print(classification_report(ptargetTest,RFCpredict))
print(accuracy_score(ptargetTest,RFCpredict))
```

```
              precision    recall  f1-score   support

           1       0.09      0.05      0.06       122
           2       0.10      0.05      0.06       124
           3       0.15      0.08      0.10       146
           4       0.12      0.05      0.07       159
           5       0.09      0.04      0.06       165
           6       0.09      0.09      0.09       191
           7       0.13      0.25      0.17       244
           8       0.10      0.28      0.14       200
           9       0.14      0.14      0.14       204
          10       0.08      0.04      0.05       132
          11       0.16      0.06      0.09       168
          12       0.11      0.06      0.07       145

   micro avg       0.11      0.11      0.11      2000
   macro avg       0.11      0.10      0.09      2000
weighted avg       0.12      0.11      0.10      2000

0.1125
```

With using PCA and three features,

use PCA data seems to be more effective than choose three features,

and use SVC model get highest accuracy score

predict offense code group

choose feature with pearson matrix

In [96]:
```
pearsonMatrix.sort_values(by='OFFENSE_CODE_GROUP',ascending=False)
```

Out[96]:

	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DIS
OFFENSE_CODE_GROUP	0.251910	1.000000	0.563958	-0.0
OFFENSE_DESCRIPTION	0.458922	0.563958	1.000000	0.0
OFFENSE_CODE	1.000000	0.251910	0.458922	0.0
UCR_PART	0.226169	0.191804	0.076092	-0.0
YEAR	0.043738	0.012072	0.017729	0.0
REPORTING_AREA	0.015461	0.011612	0.002161	0.1
DAY_OF_WEEK	-0.002096	-0.001146	0.004687	0.0
DISTRICT	0.015447	-0.002160	0.003713	1.0
STREET	0.002873	-0.005105	-0.003713	-0.0
MONTH	-0.013767	-0.005614	-0.009981	-0.0
HOUR	-0.017109	-0.022071	-0.016076	0.0
SHOOTING	-0.058043	-0.076569	-0.051014	-0.0

In [97]:
```
offenseSample=tras.sample(n=10000)
odata=offenseSample.loc[:,['UCR_PART','YEAR','REPORTING_AREA']]
otarget=offenseSample.loc[:,'OFFENSE_CODE_GROUP']
```

In [99]:
```
odata.head()
```

Out[99]:

	UCR_PART	YEAR	REPORTING_AREA
247171	2	2016	449
133387	3	2017	380
110035	2	2017	0
309231	3	2015	13
73547	2	2018	373

In [100]:
```
odata['YEAR']=LabelEncoder().fit_transform(odata['YEAR'])
```

In [101]:
```
otarget.unique()
```

Out[101]:
```
array([43, 46, 62, 22, 61, 63, 34, 65, 51, 37, 36, 31, 35, 40,  0, 41, 21,
       12,  4, 66,  6, 64,  3, 59, 32, 15, 56, 52, 24, 14, 58, 27, 42,  5,
       49, 17, 45, 55, 38, 11, 53, 20, 10, 16, 50, 57, 33, 47, 44, 19, 29,
        8, 39, 60, 54,  2, 28, 13,  7,  1, 30], dtype=int64)
```

predict offense code group with three feature and svc

In [105]:
```
odataTrain,odataTest, \
otargetTrain,otargetTest = \
train_test_split(odata,otarget,train_size=0.8)
```

In [103]:
```
offenseSVC=SVC()
```

In [104]:
```
offenseSVC
```

Out[104]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [106]:
```
offenseSVC.fit(odataTrain,otargetTrain)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: Fu
tureWarning: The default value of gamma will change from 'auto' to 'scale' in v
ersion 0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[106]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [107]:
```
offensepred=offenseSVC.predict(odataTrain)
```

In [108]:
```
print(classification_report(otargetTrain,offensepred))
print(accuracy_score(otargetTrain,offensepred))
```

	precision	recall	f1-score	support
0	0.67	0.40	0.50	204
2	0.00	0.00	0.00	3
3	0.56	0.23	0.32	22
4	0.64	0.14	0.23	127
5	1.00	0.12	0.22	33
6	0.00	0.00	0.00	32
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	2
10	0.00	0.00	0.00	26
11	0.58	0.12	0.19	94
12	0.60	0.09	0.15	34
13	0.00	0.00	0.00	3
14	0.50	0.01	0.03	75
15	0.46	0.66	0.54	430
16	0.00	0.00	0.00	10
17	0.00	0.00	0.00	14
19	0.00	0.00	0.00	43
20	0.00	0.00	0.00	19
21	1.00	0.04	0.08	50
22	0.54	0.12	0.19	160
24	0.00	0.00	0.00	2
27	1.00	0.02	0.05	81
28	0.00	0.00	0.00	6
29	0.00	0.00	0.00	4
30	0.00	0.00	0.00	2
31	0.35	0.36	0.36	442
32	0.55	0.17	0.26	261
33	0.67	0.10	0.17	21
34	0.59	0.88	0.71	638
35	0.55	0.43	0.48	279
36	0.00	0.00	0.00	11
37	0.00	0.00	0.00	42
38	0.56	0.16	0.24	32
40	0.37	0.51	0.43	576
41	0.56	0.08	0.15	119
42	0.33	0.03	0.06	90
43	0.33	0.88	0.48	936
44	0.00	0.00	0.00	14
45	0.00	0.00	0.00	15
46	0.45	0.56	0.50	409
47	0.00	0.00	0.00	14
49	0.57	0.06	0.10	70
50	0.00	0.00	0.00	7
51	0.60	0.12	0.20	102
52	0.63	0.12	0.20	248
53	0.00	0.00	0.00	19
54	0.00	0.00	0.00	4
55	0.00	0.00	0.00	40
56	0.70	0.14	0.24	134
57	0.50	0.03	0.05	36
58	0.71	0.09	0.17	127
59	0.00	0.00	0.00	29

```
          60       0.00      0.00      0.00         6
          61       0.46      0.56      0.50       407
          62       0.49      0.27      0.35       285
          63       0.47      0.53      0.50       390
          64       0.41      0.35      0.38       330
          65       0.38      0.18      0.25       155
          66       0.53      0.18      0.27       235

   micro avg       0.43      0.43      0.43      8000
   macro avg       0.33      0.15      0.16      8000
weighted avg       0.47      0.43      0.38      8000

0.431

C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [109]:

```
offensepredict=offenseSVC.predict(odataTest)
```

In [110]:
```
print(classification_report(otargetTest,offensepredict))
print(accuracy_score(otargetTest,offensepredict))
```

	precision	recall	f1-score	support
0	0.26	0.11	0.15	47
1	0.00	0.00	0.00	1
3	0.00	0.00	0.00	5
4	0.25	0.04	0.06	27
5	0.00	0.00	0.00	7
6	0.00	0.00	0.00	8
10	0.00	0.00	0.00	9
11	0.00	0.00	0.00	19
12	0.00	0.00	0.00	14
14	0.00	0.00	0.00	18
15	0.35	0.47	0.40	116
17	0.00	0.00	0.00	2
19	0.00	0.00	0.00	7
20	0.00	0.00	0.00	3
21	0.00	0.00	0.00	9
22	0.33	0.03	0.06	33
24	0.00	0.00	0.00	1
27	0.00	0.00	0.00	32
28	0.00	0.00	0.00	5
31	0.11	0.12	0.11	120
32	0.21	0.08	0.12	60
33	0.00	0.00	0.00	3
34	0.43	0.61	0.51	150
35	0.24	0.12	0.16	65
36	0.00	0.00	0.00	8
37	0.00	0.00	0.00	12
38	0.50	0.40	0.44	5
39	0.00	0.00	0.00	1
40	0.11	0.18	0.14	149
41	0.00	0.00	0.00	24
42	0.00	0.00	0.00	23
43	0.21	0.60	0.31	246
44	0.00	0.00	0.00	3
45	0.00	0.00	0.00	2
46	0.17	0.16	0.17	108
47	0.00	0.00	0.00	4
49	0.00	0.00	0.00	15
50	0.00	0.00	0.00	2
51	0.25	0.05	0.08	21
52	0.07	0.01	0.02	71
53	0.00	0.00	0.00	4
54	0.00	0.00	0.00	3
55	0.00	0.00	0.00	8
56	0.00	0.00	0.00	39
57	0.00	0.00	0.00	12
58	0.00	0.00	0.00	31
59	0.00	0.00	0.00	6
61	0.18	0.24	0.21	90
62	0.07	0.03	0.05	58
63	0.24	0.25	0.25	107
64	0.29	0.19	0.23	89
65	0.00	0.00	0.00	42

```
          66       0.22      0.07      0.11        56

   micro avg       0.22      0.22      0.22      2000
   macro avg       0.09      0.07      0.07      2000
weighted avg       0.18      0.22      0.18      2000

0.224

C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [111]:
```
opdata=offenseSample.loc[:,['OFFENSE_CODE', 'OFFENSE_CODE_GROUP','OFFENSE_DESCRI
                  'DISTRICT', 'REPORTING_AREA', 'SHOOTING', 'YEAR','DAY_OF_WEEK',
optarget=offenseSample.loc[:,'OFFENSE_CODE_GROUP']
```

In [112]:
```
opcamodel=PCA(n_components=12).fit(opdata)
```

In [113]:
```
opcamodel.explained_variance_ratio_
```

Out[113]:
```
array([5.90476449e-01, 3.90501104e-01, 1.79437532e-02, 1.00586325e-03,
       5.47510393e-05, 1.10221521e-05, 2.98730643e-06, 2.55957920e-06,
       1.13725698e-06, 2.42909671e-07, 1.29773819e-07, 8.20014352e-10])
```

In [114]:
```
opcadata=PCA(n_components=2).fit_transform(opdata)
```

In [115]:
```
opcadata.shape
```

Out[115]:
```
(10000, 2)
```

In [116]:
```
opcaDataTrain,opcaDataTest, \
opcaTargetTrain,opcaTargetTest = \
train_test_split(opcadata,optarget,train_size=0.8)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\model_selection\_sp
lit.py:2179: FutureWarning: From version 0.21, test_size will always complement
train_size unless both are specified.
  FutureWarning)
```

In [118]:
```
pcaoffenseSVC=SVC().fit(opcaDataTrain,opcaTargetTrain)
```

```
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: Fu
tureWarning: The default value of gamma will change from 'auto' to 'scale' in v
ersion 0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [119]:
```
pcapre=pcaoffenseSVC.predict(opcaDataTrain)
print(classification_report(opcaTargetTrain,pcapre))
print(accuracy_score(opcaTargetTrain,pcapre))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	212
1	1.00	1.00	1.00	1
2	1.00	1.00	1.00	1
3	1.00	0.96	0.98	24
4	1.00	1.00	1.00	121
5	1.00	1.00	1.00	33
6	1.00	0.97	0.98	29
7	1.00	1.00	1.00	1
8	0.00	0.00	0.00	1
10	1.00	1.00	1.00	29
11	1.00	1.00	1.00	91
12	1.00	1.00	1.00	28
13	1.00	1.00	1.00	3
14	1.00	1.00	1.00	76
15	1.00	1.00	1.00	431
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	11
19	0.98	1.00	0.99	41
20	1.00	1.00	1.00	17
21	1.00	1.00	1.00	47
22	1.00	1.00	1.00	149
24	1.00	1.00	1.00	3
27	1.00	1.00	1.00	84
28	1.00	1.00	1.00	8
29	1.00	1.00	1.00	3
30	1.00	0.50	0.67	2
31	0.90	1.00	0.95	437
32	0.99	0.84	0.91	247
33	1.00	0.88	0.94	17
34	0.98	1.00	0.99	631
35	1.00	0.95	0.97	281
36	1.00	1.00	1.00	13
37	1.00	0.86	0.93	36
38	0.97	1.00	0.98	30
39	1.00	1.00	1.00	1
40	1.00	1.00	1.00	573
41	0.97	1.00	0.99	116
42	1.00	0.97	0.98	94
43	1.00	1.00	1.00	961
44	1.00	1.00	1.00	15
45	1.00	1.00	1.00	15
46	1.00	1.00	1.00	417
47	1.00	1.00	1.00	15
49	0.96	0.96	0.96	71
50	1.00	1.00	1.00	7
51	1.00	0.96	0.98	102
52	0.99	1.00	1.00	252
53	1.00	0.93	0.97	15
54	1.00	1.00	1.00	6
55	1.00	1.00	1.00	39
56	1.00	1.00	1.00	146

```
          57       1.00      1.00      1.00        39
          58       1.00      1.00      1.00       122
          59       1.00      1.00      1.00        29
          60       1.00      1.00      1.00         5
          61       1.00      1.00      1.00       401
          62       1.00      1.00      1.00       277
          63       1.00      1.00      1.00       396
          64       1.00      1.00      1.00       338
          65       1.00      0.99      1.00       155
          66       1.00      1.00      1.00       246

   micro avg       0.99      0.99      0.99      8000
   macro avg       0.98      0.96      0.97      8000
weighted avg       0.99      0.99      0.99      8000

0.99

C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [120]:
```
pcapredict=pcaoffenseSVC.predict(opcaDataTest)
print(classification_report(opcaTargetTest,pcapredict))
print(accuracy_score(opcaTargetTest,pcapredict))
```

	precision	recall	f1-score	support
0	1.00	0.41	0.58	39
2	0.00	0.00	0.00	2
3	0.00	0.00	0.00	3
4	1.00	0.09	0.17	33
5	0.00	0.00	0.00	7
6	0.00	0.00	0.00	11
8	0.00	0.00	0.00	1
10	1.00	0.17	0.29	6
11	1.00	0.23	0.37	22
12	1.00	0.15	0.26	20
14	1.00	0.53	0.69	17
15	1.00	0.38	0.55	115
16	0.00	0.00	0.00	1
17	0.00	0.00	0.00	5
19	0.00	0.00	0.00	9
20	0.00	0.00	0.00	5
21	0.00	0.00	0.00	12
22	1.00	0.20	0.34	44
27	1.00	0.10	0.19	29
28	1.00	0.33	0.50	3
29	0.00	0.00	0.00	1
31	0.77	0.39	0.52	125
32	0.86	0.34	0.49	74
33	0.00	0.00	0.00	7
34	0.96	0.64	0.77	157
35	0.94	0.24	0.38	63
36	0.00	0.00	0.00	6
37	1.00	0.11	0.20	18
38	1.00	0.14	0.25	7
40	1.00	0.50	0.67	152
41	0.88	0.26	0.40	27
42	0.67	0.21	0.32	19
43	0.16	1.00	0.28	221
44	0.00	0.00	0.00	2
45	0.00	0.00	0.00	2
46	1.00	0.30	0.46	100
47	0.00	0.00	0.00	3
49	0.33	0.07	0.12	14
50	0.00	0.00	0.00	2
51	1.00	0.24	0.38	21
52	0.92	0.33	0.48	67
53	0.00	0.00	0.00	8
54	0.00	0.00	0.00	1
55	0.00	0.00	0.00	9
56	0.00	0.00	0.00	27
57	0.00	0.00	0.00	9
58	1.00	0.14	0.24	36
59	1.00	0.33	0.50	6
60	0.00	0.00	0.00	1
61	1.00	0.46	0.63	96
62	1.00	0.38	0.55	66

```
          63       1.00      0.31      0.47       101
          64       1.00      0.40      0.57        81
          65       1.00      0.21      0.35        42
          66       1.00      0.58      0.73        45

   micro avg       0.41      0.41      0.41      2000
   macro avg       0.54      0.18      0.25      2000
weighted avg       0.80      0.41      0.45      2000

0.413

C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [121]:

```
opcaGBC=GBC(max_depth=66).fit(opcaDataTrain,opcaTargetTrain)
```

In [122]:
```
opcaGBCpre=opcaGBC.predict(opcaDataTrain)
print(classification_report(opcaTargetTrain,opcaGBCpre))
print(accuracy_score(opcaTargetTrain,opcaGBCpre))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	212
1	1.00	1.00	1.00	1
2	1.00	1.00	1.00	1
3	1.00	1.00	1.00	24
4	1.00	1.00	1.00	121
5	1.00	1.00	1.00	33
6	1.00	1.00	1.00	29
7	1.00	1.00	1.00	1
8	1.00	1.00	1.00	1
10	1.00	1.00	1.00	29
11	1.00	1.00	1.00	91
12	1.00	1.00	1.00	28
13	1.00	1.00	1.00	3
14	1.00	1.00	1.00	76
15	1.00	1.00	1.00	431
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	11
19	1.00	1.00	1.00	41
20	1.00	1.00	1.00	17
21	1.00	1.00	1.00	47
22	1.00	1.00	1.00	149
24	1.00	1.00	1.00	3
27	1.00	1.00	1.00	84
28	1.00	1.00	1.00	8
29	1.00	1.00	1.00	3
30	1.00	1.00	1.00	2
31	1.00	1.00	1.00	437
32	1.00	1.00	1.00	247
33	1.00	1.00	1.00	17
34	1.00	1.00	1.00	631
35	1.00	1.00	1.00	281
36	1.00	1.00	1.00	13
37	1.00	1.00	1.00	36
38	1.00	1.00	1.00	30
39	1.00	1.00	1.00	1
40	1.00	1.00	1.00	573
41	1.00	1.00	1.00	116
42	1.00	1.00	1.00	94
43	1.00	1.00	1.00	961
44	1.00	1.00	1.00	15
45	1.00	1.00	1.00	15
46	1.00	1.00	1.00	417
47	1.00	1.00	1.00	15
49	1.00	1.00	1.00	71
50	1.00	1.00	1.00	7
51	1.00	1.00	1.00	102
52	1.00	1.00	1.00	252
53	1.00	1.00	1.00	15
54	1.00	1.00	1.00	6
55	1.00	1.00	1.00	39
56	1.00	1.00	1.00	146

```
              57       1.00      1.00      1.00        39
              58       1.00      1.00      1.00       122
              59       1.00      1.00      1.00        29
              60       1.00      1.00      1.00         5
              61       1.00      1.00      1.00       401
              62       1.00      1.00      1.00       277
              63       1.00      1.00      1.00       396
              64       1.00      1.00      1.00       338
              65       1.00      1.00      1.00       155
              66       1.00      1.00      1.00       246

   micro avg       1.00      1.00      1.00      8000
   macro avg       1.00      1.00      1.00      8000
weighted avg       1.00      1.00      1.00      8000

1.0
```

In [123]:
```
opcaGBCpredict=opcaGBC.predict(opcaDataTest)
print(classification_report(opcaTargetTest,opcaGBCpredict))
print(accuracy_score(opcaTargetTest,opcaGBCpredict))
```

	precision	recall	f1-score	support
0	0.93	0.97	0.95	39
2	0.00	0.00	0.00	2
3	0.00	0.00	0.00	3
4	0.94	0.97	0.96	33
5	0.83	0.71	0.77	7
6	0.67	0.18	0.29	11
8	0.00	0.00	0.00	1
10	1.00	0.17	0.29	6
11	0.41	0.41	0.41	22
12	1.00	0.85	0.92	20
14	1.00	1.00	1.00	17
15	1.00	1.00	1.00	115
16	0.00	0.00	0.00	1
17	0.25	0.20	0.22	5
19	0.20	0.22	0.21	9
20	0.00	0.00	0.00	5
21	0.92	1.00	0.96	12
22	0.67	0.68	0.67	44
27	0.79	0.66	0.72	29
28	1.00	0.33	0.50	3
29	1.00	1.00	1.00	1
31	0.66	0.70	0.68	125
32	0.56	0.51	0.54	74
33	0.00	0.00	0.00	7
34	0.84	0.84	0.84	157
35	0.62	0.59	0.60	63
36	0.00	0.00	0.00	6
37	0.33	0.17	0.22	18
38	0.50	0.14	0.22	7
40	0.88	0.99	0.93	152
41	0.72	0.78	0.75	27
42	0.65	0.58	0.61	19
43	0.91	1.00	0.95	221
44	0.00	0.00	0.00	2
45	0.40	1.00	0.57	2
46	0.78	0.88	0.83	100
47	0.33	0.33	0.33	3
49	0.16	0.21	0.18	14
50	0.00	0.00	0.00	2
51	0.59	0.48	0.53	21
52	0.81	0.85	0.83	67
53	1.00	0.12	0.22	8
54	1.00	1.00	1.00	1
55	1.00	0.78	0.88	9
56	1.00	0.96	0.98	27
57	0.80	0.89	0.84	9
58	0.97	0.94	0.96	36
59	0.75	0.50	0.60	6
60	0.00	0.00	0.00	1
61	0.98	1.00	0.99	96
62	0.98	0.98	0.98	66

```
          63       0.98      0.99      0.99       101
          64       0.96      0.99      0.98        81
          65       0.93      0.88      0.90        42
          66       0.86      0.96      0.91        45

   micro avg       0.83      0.83      0.83      2000
   macro avg       0.63      0.57      0.58      2000
weighted avg       0.82      0.83      0.82      2000

0.833

C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Benedict Arora\Anaconda3\lib\site-packages\sklearn\metrics\classificat
ion.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

predict offense_code_group

With using PCA and GradientBoostingClassifier,

accuracy score is 89%,and some class's f1_score reach 1.00

So,best way may be using PCA and GradientBoostingClassifier

In [ ]:

```
```