|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Crime in India

'PG PROGRAM IN DATA SCIENCE, MACHINE LEARNING AND NEURAL NETWORKS'

**PREPARED BY:**

**SHEFALI MEVADA**

**BATCH NO: 1826**

##### In this blog-post, I will go through the whole process of creating a machine learning model on the Crime in India dataset, In which It provides Property stolen and recovered cases information of the cities of India. Cases of property stolen, recovered and values of property stolen and recovered.

##### Crime in India

This dataset contains complete information about various aspects of crimes happened in India from 2001. There are many factors that can be analyzed from this dataset. In which I choose Property stolen and recovered case dataset.

##### 

**Importing the Libraries**

# linear algebra  
**import** **numpy** **as** **np**   
  
# data processing  
**import** **pandas** **as** **pd**

# data visualization

**import seaborn as sns  
from matplotlib.pyplot import pyplot as plt**

# Algorithms

##### from scipy.stats import zscore

##### from sklearn.metrics import accuracy\_score

##### from sklearn.metrics import confusion\_matrix

##### from sklearn.metrics import classification\_report

##### from sklearn.preprocessing import LabelEncoder

##### from sklearn.preprocessing import StandardScaler

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

##### from sklearn.linear\_model import LinearRegression

##### from sklearn.model\_selection import cross\_val\_score

##### from sklearn.metrics import r2\_score

##### from sklearn.ensemble import RandomForestRegressor

##### from sklearn.ensemble import AdaBoostRegressor

##### from sklearn.linear\_model import SGDRegressor

##### from sklearn.model\_selection import GridSearchCV

##### import warnings

##### warnings.filterwarnings('ignore')

# Getting the Data

df=pd.read\_csv('crimeinindia.csv')

# Data Exploration/Analysis

df.info()

**<class 'pandas.core.frame.DataFrame'>**

**RangeIndex: 2449 entries, 0 to 2448**

**Data columns (total 8 columns):**

**# Column Non-Null Count Dtype**

**--- ------ -------------- -----**

**0 Area\_Name 2449 non-null object**

**1 Year 2449 non-null int64**

**2 Group\_Name 2449 non-null object**

**3 Sub\_Group\_Name 2449 non-null object**

**4 Cases\_Property\_Recovered 2449 non-null int64**

**5 Cases\_Property\_Stolen 2449 non-null int64**

**6 Value\_of\_Property\_Recovered 2449 non-null int64**

**7 Value\_of\_Property\_Stolen 2449 non-null int64**

**dtypes: int64(5), object(3)**

**memory usage: 153.2+ KB**

**It has 2449** **examples and 8 features + the target variable (Cases\_Property\_Recovered)**. 5 of the features are integers and 3 are objects. Below I have listed the features with a short description:

Area\_Name: Name of Cities In India

Year : Year of Case

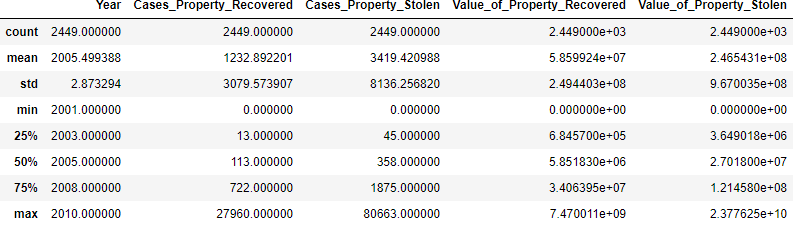
Group\_Name : Name of Group

Sub\_Group\_Name : Name of Sub Group

Cases\_Property\_Recovered : Number of Recovered Property CasesCases\_Property\_Stolen : Number of Stolen Property CasesValue\_of\_Property\_Recovered : Value of Recovered Property CasesValue\_of\_Property\_Stolen : Value of Stolen Property Cases

**Summary statistics**

**df.describe()**

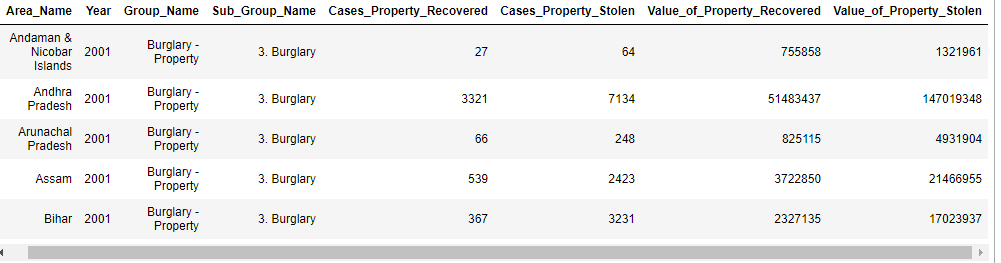
****

Here, In above Image We Describing the Datasets. we are determining mean, standard deviation, minimum and maximum value of each column, It helps Us further in Data Cleaning.

From Above observation we can see that The minimum cases of property recovered is 0 and maximum is 80663.

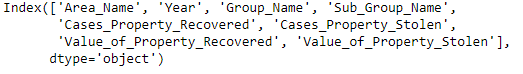
Mean is greater than median that’s why target variable is right skewed.

**df.head()**

****

From the above table, We can see the first five row of data. As per the data there is no null values in any coolumns.

**df.columns**

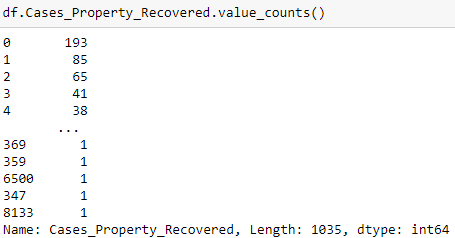
****

Now we are checking for the Unique values

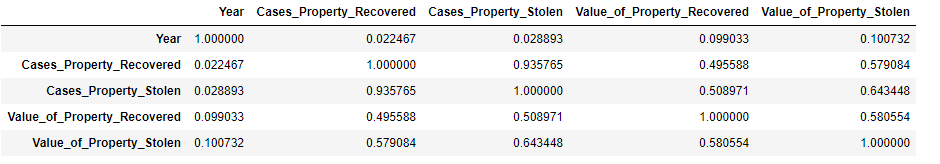
****

Here we check the Unique values of "Cases\_Property\_Recovered" column.

Now we are checking for the value counts of "Cases\_Property\_Recovered" column

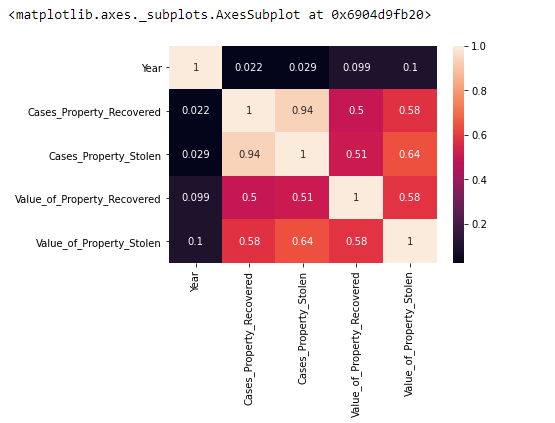
****

Now we are checking the Correlation between all the columns

****

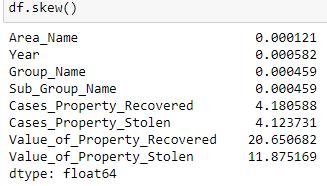
we can see the correlation between all the columns of dataset. For better visualization let us heatmap of this correlation chart.

Describing Correlation using heatmap



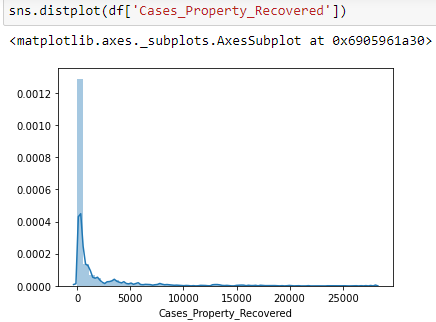
In this above Chart of Correlation using heatmap, All the columns of datasets is positively correlated with the target column. we can observe cases of property recovered is highly negatively correlated with year.

Now let's check the Skewness for Dataset of all columns



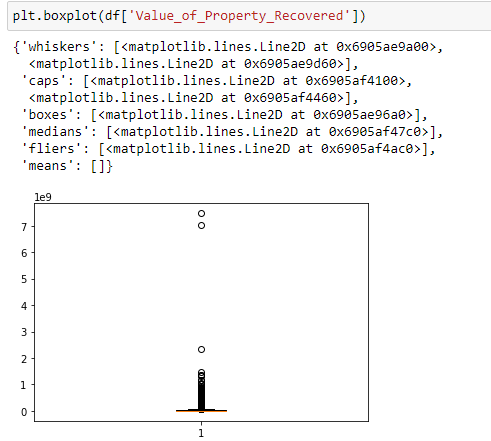
Here we get skewness for each and every columns. Skewness threshold is taken is +/-3.

Normal Distribution Curve:



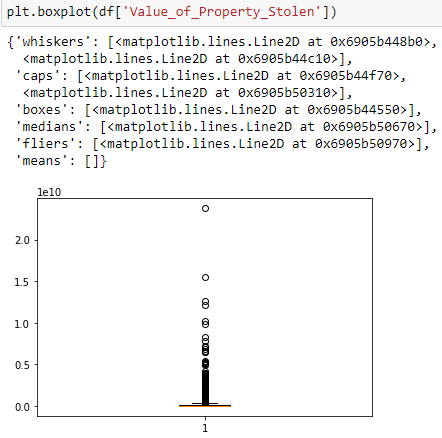
The data of the column is not Normalised. The building blocks is out of the normalised curve.

Checking skewness of "Value\_of\_Property\_Recovered" column with using boxplot.



As per the above Image, The Outcome of "Value\_of \_Property\_Recovered" column shows that there is many Outliers. The Data is Skewed.

Now let's check the skewness graph for "Value\_of\_Property\_Stolen" column using boxplot library



Here we can see in above image, the outcome of this column "Value\_of\_Property\_Stolen" data is skewed.

**Using Pairplot for checking Outliers of all the columns**

we have only 8 columns so we can go for pairplot graph and represent the outliers of columns together.

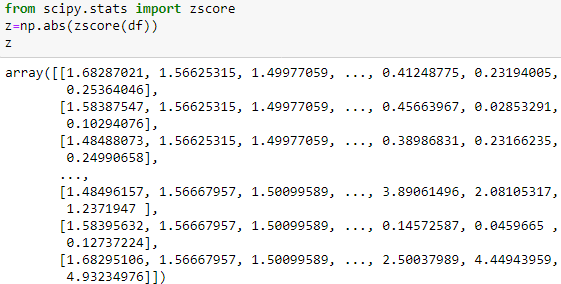
<seaborn.axisgrid.PairGrid at 0x6905bb8700>



From the above outcome we can see that columns ["Area\_Name","Year","Group\_Name","Sub\_Group\_Name"] has less outliers, columns ["Cases\_Property\_Recovered","Cases\_Property\_Stolen", "Value\_of\_Property\_Recovered", "Value\_of\_Property\_Stolen"] has more outliers.

The Pair plot shows that the data is skewed.

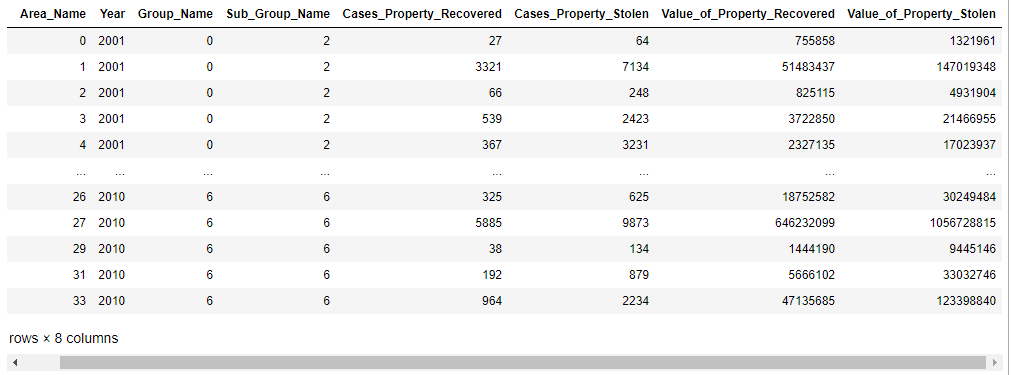
Removing Outliers With Using z-score method,



threshold=3

new\_df=df[(z<3).all(axis=1)]

new\_df



Now there is 2350 rows and 8 columns.

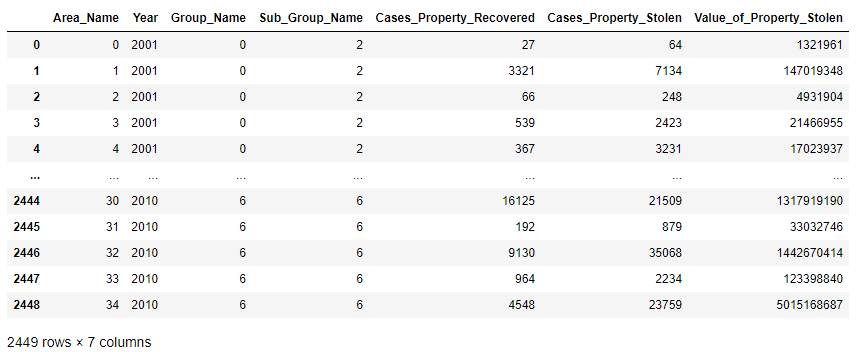
**Data Preprocessing**

**Data Cleaning**

I will drop the "Value\_of\_Property\_Recovered" from the set, because it does not contribute the dataset.

df.drop('Value\_of\_Property\_Recovered',axis=1,inplace=True)

df



as per the above image, we can see the reduce of rows and columns. Now it has 2449 rows and 7 columns.

# Building Machine Learning Models

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

x=new\_df.drop("Cases\_Property\_Recovered",axis=1)

y=new\_df["Cases\_Property\_Recovered"]

**StandardScaler:**

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

scaledX=sc.fit\_transform(x)

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

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import r2\_score

best\_rstate=0

accu=0

for i in range (30,200):

x\_train,x\_test,y\_train,y\_test=train\_test\_split(scaledX,y,test\_size=.25,random\_state=i)

mod=LinearRegression()

mod.fit(x\_train,y\_train)

y\_pred=mod.predict(x\_test)

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

if tempaccu>accu:

accu=tempaccu

best\_rstate=i

print (f"best accuracy {accu\*100} found on random\_state{best\_rstate}")

**best accuracy 94.43439385654389 found on random\_state56**

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(scaledX,y,test\_size=.25,random\_state=104)

**Since the Outcome has real values we will use LinearRegression algorithm**

**LinearRegression :**

from sklearn.linear\_model import LinearRegression

LR=LinearRegression()

LR.fit(x\_train,y\_train)

y\_pred=LR.predict(x\_test)

r2score=r2\_score(y\_test,y\_pred)

cvscore=cross\_val\_score(LinearRegression(),x\_train,y\_train,cv=5).mean()

print(f"Accuracy={r2score\*100}, Cross\_val\_score= {cvscore\*100}, & difference= {(r2score\*100)-(cvscore\*100)}")

**Accuracy=92.27908983997469, Cross\_val\_score= 91.11095319452639, & difference= 1.1681366454482998**

**RandomForestRegressor :**

from sklearn.ensemble import RandomForestRegressor

RF=RandomForestRegressor()

RF.fit(x\_train,y\_train)

y\_pred=RF.predict(x\_test)

r2score=r2\_score(y\_test,y\_pred)

cvscore=cross\_val\_score(RandomForestRegressor(),x\_train,y\_train,cv=5).mean()

print(f"Accuracy={r2score\*100}, Cross\_val\_score= {cvscore\*100}, & difference= {(r2score\*100)-(cvscore\*100)}")

**Accuracy=96.57681007599784, Cross\_val\_score= 93.93010474506622, & difference= 2.646705330931624**

**AdaBoostRegressor :**

from sklearn.ensemble import AdaBoostRegressor

ADB=AdaBoostRegressor()

ADB.fit(x\_train,y\_train)

y\_pred=ADB.predict(x\_test)

r2score=r2\_score(y\_test,y\_pred)

cvscore=cross\_val\_score(AdaBoostRegressor(),x\_train,y\_train,cv=5).mean()

print(f"Accuracy={r2score\*100}, Cross\_val\_score= {cvscore\*100}, & difference= {(r2score\*100)-(cvscore\*100)}")

**Accuracy=90.37863282005715, Cross\_val\_score= 88.82595837626361, & difference= 1.552674443793535**

**SGDRegressor:**

from sklearn.linear\_model import SGDRegressor

SGD=SGDRegressor()

SGD.fit(x\_train,y\_train)

y\_pred=LR.predict(x\_test)

r2score=r2\_score(y\_test,y\_pred)

cvscore=cross\_val\_score(SGDRegressor(),x\_train,y\_train,cv=5).mean()

print(f"Accuracy={r2score\*100}, Cross\_val\_score= {cvscore\*100}, & difference= {(r2score\*100)-(cvscore\*100)}")

**Accuracy=92.27908983997469, Cross\_val\_score= 91.16323272136569, & difference= 1.115857118609**

parameters= {"max\_iter":[500,700,900,1100,1200,1300,1400,1500],

"alpha":[0.0001,0.001,0.01,0.1,1,10,100],

"penalty":["elasticnet","l1","l2"]}

**GridSearchCV :**

from sklearn.model\_selection import GridSearchCV

GCV=GridSearchCV(SGDRegressor(),parameters,cv=5,scoring='r2')

GCV.fit(x\_train,y\_train)

GCV.best\_estimator\_

GCV\_pred=GCV.best\_estimator\_.predict(x\_test)

print("Final accuracy:", r2\_score (y\_test,GCV\_pred)\*100)

**Final accuracy: 92.30125840746307**

THANK YOU