A picture containing shape

Description automatically generated

**Outlier Treatments**

**Instructions**:

Please share your answers filled inline in the word document. Submit code files wherever applicable.

Please ensure you update all the details:

**Name: \_\_\_\_\_\_\_RAJU BOTTA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Batch Id: \_\_\_\_\_\_\_05102021\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Topic: Data Pre-Processing**

**Problem Statement:**

Most of the datasets have extreme values or exceptions in their observations. These values affect the predictions (Accuracy) of the model in one way or the other, removing these values is not a very good option. For these types of scenarios, we have various techniques to treat such values.

Refer: <https://360digitmg.com/mindmap-data-science>

1. Prepare the dataset by performing the preprocessing techniques, to treat the outliers.

A picture containing shape, arrow

Description automatically generated**

**Hints:**

For each assignment, the solution should be submitted in the below format

1. Work on each feature to create a data dictionary as displayed in the image displayed below:Table

   Description automatically generated
2. Hint: Boston dataset is publicly available. Refer to Boston.csv file.
3. Research and perform all possible steps for obtaining solution
4. All the codes (executable programs) should execute without errors
5. Code modularization should be followed
6. Each line of code should have comments explaining the logic and why you are using that function
7. Detailed explanation of your approach is mandatory

|  |
| --- |
| **Name of Feature             |     Description          |   Type                          |  Relevance**  crim           -  per capita crime rate by town            |  Float (continuous)      |  relevant  zn            -       proportion of residential land zoned for lots over 25,000 sq.ft    | Float                           | relevant  indus        -               proportion of non-retail business acres per town              | Float                            | relevant  chas         -              Charles River dummy variable | Float                              | relevant |
| nox           -             nitric oxides concentration (parts per 10 million)               | Float                              | relevant |
| rm            -              average number of rooms per dwelling                          | Float                              | relevant |
| age          -            proportion of owner-occupied units built prior to 1940       |  Float                          |  relevant |
| dis            -               weighted distances to five Boston employment centers  | Float                       | relevant |
| rad          -      index of accessibility to radial highways               | int                                    | relevant |
| tax           -     full-value property-tax rate per $10,000                              | int                                    |   relevant |
| ptratio      -     pupil-teacher ratio by town | float                                  |   relevant |
| black       -     1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town |  float                                |   relevant |
| lstat        -  % lower status of the population                              -                float                                 |  relevant |

import pandas as pd

import numpy as np

import seaborn as sns

df = pd.read\_csv(r'D:\DataSets\boston\_data.csv')

df

df.shape

**#let's find outliers, and data is normally distributed or not**

sns.boxplot(df.crim)

sns.boxplot(df.zn)

sns.boxplot(df.indus)

sns.boxplot(df.nox) # no outliers

sns.boxplot(df.rm) #No outliers

sns.boxplot(df.age) #No Outliers

sns.boxplot(df.dis)

sns.boxplot(df.rad)

sns.boxplot(df.tax) #no outliers

sns.boxplot(df.ptratio)

sns.boxplot(df.black)

sns.boxplot(df.lstat)

sns.boxplot(df.medv)

**# RRR Technique to treat outliers**

**#outlier treatment for crim**

IQR = df['crim'].quantile(0.75)-df['crim'].quantile(0.25)

lower\_limit = df['crim'].quantile(0.25)-(IQR)\*1.5

upper\_limit = df['crim'].quantile(0.75)+(IQR)\*1.5

**#1 remove or Trimming technic**

outliers\_df = np.where(df['crim']>upper\_limit,True, np.where(df['crim']<lower\_limit,True, False))

df\_trimmed = df.loc[~(outliers\_df),]

df\_trimmed.shape

#explore outliers in the trimmed data set

sns.boxplot(df\_trimmed.crim)

plt.title('boxplot')

**# 2. replace**

df['df\_replace'] = pd.DataFrame(np.where(df['crim']>upper\_limit,upper\_limit, np.where(df['crim']<lower\_limit,lower\_limit,df['crim'])))

sns.boxplot(df['df\_replace'])

plt.title('box plot')

plt.show()

**#3. retain or winsorization**

from feature\_engine.outliers import Winsorizer

winsor = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('crim'))

df\_t = winsor.fit\_transform(df[['crim']])

df\_t

sns.boxplot(df\_t.crim)

plt.title('Box Plot')

plt.show

**#outlier treatment for zn**

**#Finding IQR for zn**

IQR1 = df['zn'].quantile(0.75)-df['zn'].quantile(0.25)

lower\_limit1 = df['zn'].quantile(0.25)-(IQR1)\*1.5

upper\_limit1 = df['zn'].quantile(0.75)+(IQR1)\*1.5

**#remove**

outliers\_df1 = np.where(df['zn']>upper\_limit1,True, np.where(df['zn']<lower\_limit1,True, False))

df\_trimmed1 = df.loc[~(outliers\_df1),]

df\_trimmed1.shape

sns.boxplot(df\_trimmed1.zn)

plt.title('boxplot')

**#replace**

df['df\_replace1'] = pd.DataFrame(np.where(df['zn']>upper\_limit1,upper\_limit1, np.where(df['zn']<lower\_limit1,lower\_limit1,df['zn'])))

sns.boxplot(df['df\_replace1'])

plt.title('box plot')

plt.show()

**#retain or winsorize**

winsor1 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('zn'))

df\_t = winsor1.fit\_transform(df[['zn']])

df\_t

sns.boxplot(df\_t.zn)

plt.title('Box Plot')

plt.show

**#outlier treatment for indus**

#IQR for indus

IQR2 = df['indus'].quantile(0.75)-df['indus'].quantile(0.25)

lower\_limit2 = df['indus'].quantile(0.25)-(IQR2)\*1.5

upper\_limit2= df['indus'].quantile(0.75)+(IQR2)\*1.5

**#remove**

outliers\_df2 = np.where(df['indus']>upper\_limit2,True, np.where(df['indus']<lower\_limit2,True, False))

df\_trimmed2 = df.loc[~(outliers\_df2),]

df\_trimmed2.shape

sns.boxplot(df\_trimmed2.indus)

plt.title('boxplot')

#**replace**

df['df\_replace2'] = pd.DataFrame(np.where(df['indus']>upper\_limit2,upper\_limit2, np.where(df['indus']<lower\_limit2,lower\_limit2,df['indus'])))

sns.boxplot(df['df\_replace2'])

plt.title('box plot')

plt.show()

**#retain or winsorize**

winsor2 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('indus'))

df\_t = winsor2.fit\_transform(df[['indus']])

df\_t

sns.boxplot(df\_t.indus)

plt.title('Box Plot')

plt.show

**#outlier treatment for dis**

#IQR

IQR3 = df['dis'].quantile(0.75)-df['dis'].quantile(0.25)

lower\_limit3 = df['dis'].quantile(0.25)-(IQR3)\*1.5

upper\_limit3= df['dis'].quantile(0.75)+(IQR3)\*1.5

**#Remove**

outliers\_df3 = np.where(df['dis']>upper\_limit3,True, np.where(df['dis']<lower\_limit3,True, False))

df\_trimmed3 = df.loc[~(outliers\_df3),]

df\_trimmed3.shape

sns.boxplot(df\_trimmed3.dis)

plt.title('boxplot')

**#replace**

df['df\_replace3'] = pd.DataFrame(np.where(df['dis']>upper\_limit3,upper\_limit3, np.where(df['dis']<lower\_limit3,lower\_limit3,df['dis'])))

sns.boxplot(df['df\_replace3'])

**#retain or winsorize**

winsor3 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('dis'))

df\_t = winsor3.fit\_transform(df[['dis']])

df\_t

sns.boxplot(df\_t.dis)

plt.title('Box Plot')

plt.show

**#outlier treatment for rad**

**#IQR**

IQR4 = df['rad'].quantile(0.75)-df['rad'].quantile(0.25)

lower\_limit4 = df['rad'].quantile(0.25)-(IQR4)\*1.5

upper\_limit4= df['rad'].quantile(0.75)+(IQR4)\*1.5

**#remove**

outliers\_df4 = np.where(df['rad']>upper\_limit4,True, np.where(df['rad']<lower\_limit4,True, False))

df\_trimmed4 = df.loc[~(outliers\_df4),]

df\_trimmed4.shape

sns.boxplot(df\_trimmed4.rad)

plt.title('boxplot')

#**replace**

df['df\_replace4'] = pd.DataFrame(np.where(df['rad']>upper\_limit4,upper\_limit4, np.where(df['rad']<lower\_limit4,lower\_limit4,df['rad'])))

sns.boxplot(df['df\_replace4'])

**#retain or winsorize for rad**

winsor4 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('rad'))

df\_t = winsor4.fit\_transform(df[['rad']])

df\_t

sns.boxplot(df\_t.rad)

plt.title('Box Plot')

plt.show

**#outlier treatment ptratio**

**#IQR**

IQR5 = df['ptratio'].quantile(0.75)-df['ptratio'].quantile(0.25)

lower\_limit5 = df['ptratio'].quantile(0.25)-(IQR5)\*1.5

upper\_limit5= df['ptratio'].quantile(0.75)+(IQR5)\*1.5

**#remove**

outliers\_df5 = np.where(df['ptratio']>upper\_limit5,True, np.where(df['ptratio']<lower\_limit5,True, False))

df\_trimmed5 = df.loc[~(outliers\_df5),]

df\_trimmed5.shape

sns.boxplot(df\_trimmed5.ptratio)

plt.title('boxplot')

**#replace**

df['df\_replace5'] = pd.DataFrame(np.where(df['ptratio']>upper\_limit5,upper\_limit5, np.where(df['ptratio']<lower\_limit5,lower\_limit5,df['ptratio'])))

sns.boxplot(df['df\_replace5'])

**#retain or winsorize for rad**

winsor5 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('ptratio'))

df\_t = winsor5.fit\_transform(df[['ptratio']])

df\_t

sns.boxplot(df\_t.ptratio)

plt.title('Box Plot')

plt.show

**#outlier treatment for black**

**#IQR**

IQR6 = df['black'].quantile(0.75)-df['black'].quantile(0.25)

lower\_limit6 = df['black'].quantile(0.25)-(IQR)\*1.5

upper\_limit6 = df['black'].quantile(0.75)+(IQR)\*1.5

**#remove**

outliers\_df6 = np.where(df['black']>upper\_limit6,True, np.where(df['black']<lower\_limit6,True, False))

df\_trimmed6 = df.loc[~(outliers\_df6),]

df\_trimmed6.shape

sns.boxplot(df\_trimmed6.black)

plt.title('boxplot')

**#replace**

df['df\_replace6'] = pd.DataFrame(np.where(df['black']>upper\_limit6,upper\_limit6, np.where(df['black']<lower\_limit6,lower\_limit6,df['black'])))

sns.boxplot(df['df\_replace6'])

**#retain or winsorize for rad**

winsor6 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('black'))

df\_t = winsor6.fit\_transform(df[['black']])

df\_t

sns.boxplot(df\_t.black)

plt.title('Box Plot')

plt.show

**#outlier treatment for black**

**#IQR**

IQR7 = df['istat'].quantile(0.75)-df['Istat'].quantile(0.25)

lower\_limit7 = df['Istat'].quantile(0.25)-(IQR)\*1.5

upper\_limit7 = df['Istat'].quantile(0.75)+(IQR)\*1.5

#**remove**

outliers\_df7 = np.where(df['Istat']>upper\_limit7,True, np.where(df['Istat']<lower\_limit7,True, False))

df\_trimmed7 = df.loc[~(outliers\_df7),]

df\_trimmed7.shape

sns.boxplot(df\_trimmed7.black)

plt.title('boxplot')

#**replace**

df['df\_replace7'] = pd.DataFrame(np.where(df['Istat']>upper\_limit7,upper\_limit7, np.where(df['Istat']<lower\_limit7,lower\_limit7,df['Istat'])))

sns.boxplot(df['df\_replace7'])

**#retain or winsorize for rad**

winsor7 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('Istat'))

df\_t = winsor7.fit\_transform(df[['Istat']])

df\_t

sns.boxplot(df\_t.Istat)

plt.title('Box Plot')

plt.show

**#outlier treatment for medv**

**#IQR**

IQR8 = df['medv'].quantile(0.75)-df['medv'].quantile(0.25)

lower\_limit8 = df['medv'].quantile(0.25)-(IQR)\*1.5

upper\_limit8 = df['medv'].quantile(0.75)+(IQR)\*1.5

**#remove**

outliers\_df8 = np.where(df['medv']>upper\_limit8,True, np.where(df['medv']<lower\_limit8,True, False))

df\_trimmed8 = df.loc[~(outliers\_df8),]

df\_trimmed8.shape

sns.boxplot(df\_trimmed8.black)

plt.title('boxplot')

**#replace**

df['df\_replace8'] = pd.DataFrame(np.where(df['medv']>upper\_limit8,upper\_limit8, np.where(df['medv']<lower\_limit8,lower\_limit8,df['medv'])))

sns.boxplot(df['df\_replace8'])

**#retain or winsorize for rad**

winsor8 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('medv'))

df\_t = winsor8.fit\_transform(df[['medv']])

df\_t

sns.boxplot(df\_t.medv)

plt.title('Box Plot')

plt.show

**#outlier treatment for lstat**

**#IQR**

IQR9 = df['lstat'].quantile(0.75)-df['lstat'].quantile(0.25)

lower\_limit9 = df['lstat'].quantile(0.25)-(IQR)\*1.5

upper\_limit9 = df['lstat'].quantile(0.75)+(IQR)\*1.5

**#remove**

outliers\_df9 = np.where(df['lstat']>upper\_limit9,True, np.where(df['lstat']<lower\_limit9,True, False))

df\_trimmed9 = df.loc[~(outliers\_df9),]

df\_trimmed9.shape

sns.boxplot(df\_trimmed9.lstat)

plt.title('boxplot')

**#replace**

df['df\_replace9'] = pd.DataFrame(np.where(df['lstat']>upper\_limit9,upper\_limit9, np.where(df['lstat']<lower\_limit9,lower\_limit9,df['lstat'])))

sns.boxplot(df['df\_replace9'])

**#retain or winsorize for rad**

winsor9 = Winsorizer(capping\_method='iqr',tail='both',fold=1.5, variables=('lstat'))

df\_t = winsor9.fit\_transform(df[['lstat']])

df\_t

sns.boxplot(df\_t.lstat)

plt.title('Box Plot')

plt.show

**Conclusion**: In the above problem outlier analysis has done using 3 technics remove, replace and retain or winsorize based on IQR (Inter Quartile Range).