**EDA Basics**

|  |  |
| --- | --- |
| top 5 | data.head() or data.head(end\_index\_value) |
| bottom 5 | data.tail() |
| random values 5 | data.sample(5) |
| shape (no. of rows,col) | data.shape |
| Name of columns | data.columns |
| duplicate value count | data[‘col\_name’].value\_counts() |
| describe (quantile) | data.describe() |
| columns data type | data.dtypes |
| range index | data.index |
| Information of data | data.info() |
| Display full dataset; Full Table | pd.set\_option('display.max\_columns',None)  pd.set\_option('display.max\_rows',None) |

**EDA Advance**

**EDA and DATA Cleaning Techniques**

**Data Cleaning Technique 1**

|  |  |
| --- | --- |
| Load Dataset | data=pd.read\_csv(‘dataset\_name.csv’) |
| To Display Full Table | pd.set\_option('display.max\_columns',None)  pd.set\_option('display.max\_rows',None) |
| Null values by column wise | data.isnull().sum() |
| Null values Total | data.isnull().sum().sum() |
| Null values Percentage (null\_per) | data.isnaull().sum()\*100/len(data) |
| Heatmap | plt.figure(figsize=((5,5))  sns.heatmap(data)  plt.show() |
| graph or plot size | plt.figure(figsize=((weidth,length)) |
| Null values above 20% | drop\_col=null\_per[null\_per>20].keys()  drop\_col and null\_per are variables |
| drop Null values by row | data.dropna() or data.dropna(axis=0) |
| drop Null values by column | data.dropna(axis=1) |
| display only Numerical columns | data3.select\_dtypes(include=['int64','float64']).keys()  **or**  data3.select\_dtypes(include=['int64','float64']).columns |
| Distplot before/after comparison by removing NaN values  Legends mention | sns.distplot(data[‘col’],) 🡪 original data (with NaN values)  sns.distplot(data3[‘col’]) 🡪 modified data (No NaN values)  plt.legend(['data','data3']) (or)  plt.legend(['org','new'])  plt.show() |
| Subplots | plt.sumplot(rows,columns,location) |
| for loop to get both data to  Auto plot all the columns  in distplot ;  here mentioning figure size  and subplot for each plot | plt.figure(figsize=(25,25))  for i, var in enumerate(list1): 🡪list1=names of columns in list  plt.subplot(4,3,i+1)  sns.distplot(data[var], bins = 20) 🡪 bin= bars  sns.distplot(data3[var], bins = 20)  plt.legend(['data','data3'] |

**Data Cleaning Technique 2**

|  |  |
| --- | --- |
| all column names into list | b=list(data1.columns) |
| for loop to fill NaN values  (b=list of all column names)  mean for numerical  median for numerical with outliers  mod for object/string | for i in b:  if data1[i].dtypes=='object' :  data1[i]=data1[i].fillna(data1[i].mod()[0])  else :  data1[i]=data1[i].fillna(data1[i].median()) |
| Extracting numerical columns for outliers | Numerical = []  for i in dataset :  if (dataset[i].dtypes == 'int64') or (dataset[i].dtypes == 'float64') :  Numerical.append(i) |
| for loop to get all box plots to see the outliers of numerical columns  Outlieres | for i in col:  if(dataset[i].dtypes == 'int64' or dataset[i].dtypes == 'float64'):  sns.boxplot(dataset[i])  plt.xlabel(i)  plt.ylabel('count')  plt.show() |
| Outliers  Using (or) | q1 = dataset.quantile(0.25)  q3 = dataset.quantile(0.75)  iqr = q3 - q1  dataset = dataset[(~((dataset < q1 - 1.5 \* iqr) | (dataset > q3 + 1.5 \* iqr)))] |
| Using (and) | dataset3 = dataset[(dataset['cgpa'] > lower) & ( dataset['cgpa'] < upper)] |

**Google Colab Link: -** <https://colab.research.google.com/drive/1Y4WfFKghcdtyaKQUeZ7XHK5udVxBrDNs?usp=sharing>

For outliers open Linear Regression Handson rohith google colab

**Data Cleaning Technique 3 (Z-Score)**

**Outliers Trimming**

|  |  |
| --- | --- |
| Limits in Outlier using Z-Score  mean – 3 \* standard deviation  mean + 3 \* standard deviation | upper = dataset['column\_name'].mean() + 3 \* dataset['col'].std()  lower = dataset['column\_name'].mean() - 3 \* dataset['col'].std() |
| Trimming the outlier  handling the outlier | dataset1 = dataset[(dataset['column\_name'] > lower) & (dataset['column\_name'] < upper)] |
| 1. Automatically finds the Outlier column  2. Apply required method to handle  accordingly  3. (Z-Score method for Normal distribution and old or Normal handling method for Skew distribution)  4. Handle the Outliers of each column we give.  5. Returns the dataset.  col = all the column names in int and float dtype and excluded the output column  col = dataset.columns or  col = list(dataset.columns) | normal\_dist = []  L\_skew\_dist = []  R\_skew\_dist = []  for i in col :  if (dataset[i].skew() > -0.5) and (dataset[i].skew() < 0.5) :  print('Normal dist')  dataset\_1 = dataset[(dataset[i] > lower) & (dataset[i] < upper)]  normal\_dist.append(i)  elif (dataset[i].skew() > -1) and (dataset[i].skew()< -0.5) :  print( 'left skew')  dataset\_2 = dataset[~((dataset[i] < Lower) | (dataset[i] > Upper))]  L\_skew\_dist.append(i)  elif (dataset[i].skew() > 0.5) and (dataset[i].skew() < 1) :  print( 'right skew')  dataset\_3 = dataset[~((dataset[i] < Lower) | (dataset[i] > Upper))]  R\_skew\_dist.append(i)  else :  print('Error') |

**Capping Technique**

|  |  |
| --- | --- |
| Normal Distribution  used Z-Score Method  capping means replacing the outliers with upper and lower limits accordingly | def zscore\_capping(col):  for i in col :  if (legit[i].skew() > -0.5) and (legit[i].skew() < 0.5) :  upper = legit[i].mean() + 3 \* legit[i].std()  lower = legit[i].mean() - 3 \* legit[i].std()  legit[i] = np.where(legit[i] > upper, upper, np.where(legit[i] < lower, lower, legit[i])) t |
| Skew Distribution  IQR method for both right skew and left skew methods | def iqr\_capping(col) :  for i in col :  if (-1 < legit[i].skew() < -0.5) or (0.5 < legit[i].skew() < 1):  Q1 = legit[i].quantile(0.25)  Q3 = legit[i].quantile(0.75)  IQR = Q3 - Q1  upp = Q3 + 1.5 \* IQR  low = Q1 - 1.5 \* IQR  legit[i] = np.where(legit[i] > upp, upp, np.where(legit[i] < low, low, legit[i])) |

Google Colab File: -

<https://colab.research.google.com/drive/1o6jjNvXPRg2ybnuB2UrAB6yjNN1W3Coj?usp=sharing>

# Assuming you have a DataFrame named 'df'

**# Save the DataFrame to a CSV file**

df.to\_csv('output\_dataset.csv', index=False)

'output\_dataset.csv' 🡪 location of the dataset to save, but (for google colab just copy the path of the uploaded dataset and paste it here.

**# Save the DataFrame to an Excel file**

df.to\_excel('output\_dataset.xlsx', index=False)