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

In [ ]:
```
df = pd.read_csv("D:\MIT ADT\Third Year - Sem 2\ML LAB\Assign 3/adult.csv")
```

info, describe, head, tail, coorelation

In [ ]:
```
df.shape
```

Out[ ]:
```
(48842, 15)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              48842 non-null  int64 
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64 
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64 
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64 
 11  capital-loss     48842 non-null  int64 
 12  hours-per-week   48842 non-null  int64 
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

In [ ]:
```
df.head()
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K

In [ ]:
```
df.tail()
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

In [ ]:
```
df.describe()
```

Out[ ]:

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

In [ ]:
```
# plt.matshow(df.corr(), cmap='winter')

df.isna().sum() #returns true false if na
```

Out[ ]:
```
age               0
workclass         0
fnlwgt            0
education         0
educational-num   0
marital-status    0
occupation        0
relationship      0
race              0
gender            0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [ ]:
```
df.duplicated().sum() #returns true false (52 before drop)
```

Out[ ]:
```
52
```

In [ ]:
```
df = df.drop_duplicates()
```

In [ ]:
```
df.isin(['?']).sum() #we need to deal with the non num missing values
```

Out[ ]:
```
age                  0
workclass         2795
fnlwgt               0
education            0
educational-num      0
marital-status       0
occupation        2805
relationship         0
race                 0
gender               0
capital-gain         0
capital-loss         0
hours-per-week       0
native-country     856
income               0
dtype: int64
```

Handling missing values

1. leave as it is 2.fill the missing values
2. drop missing values

In [ ]:
```
df = df.replace('?', np.nan)
```

In [ ]:
```

```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
age                  0
workclass         2795
fnlwgt               0
education            0
educational-num      0
marital-status       0
occupation        2805
relationship         0
race                 0
gender               0
capital-gain         0
capital-loss         0
hours-per-week       0
native-country     856
income               0
dtype: int64
```

drop na -> drop axis

In [ ]:
```
temp = pd.DataFrame({
    "name":['ABC', 'PQR', np.nan],
    "rollno": [1,np.nan,3]
})
```

In [ ]:
```
temp.fillna(method='bfill', inplace=True)
```

```
C:\Users\nilesh\AppData\Local\Temp\ipykernel_12076\221866894.py:1: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a fu
ture version. Use obj.ffill() or obj.bfill() instead.
  temp.fillna(method='bfill', inplace=True)
```

In [ ]:
```
temp
```

Out[ ]:

	name	rollno
0	ABC	1.0
1	PQR	3.0
2	NaN	3.0

In [ ]:
```
#simple inputer replace mean, mostfreq, fixed value
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy = 'most_frequent', missing_values = np.nan)
```

In [ ]:
```
df['workclass'] = imputer.fit_transform(df[['workclass']]).ravel() #ravel 2d to 1d array  #fit_tranform changes the null values with strategy in the i
```

In [ ]:
```
df['occupation'] = imputer.fit_transform(df[['occupation']]).ravel()
df['native-country'] = imputer.fit_transform(df[['native-country']]).ravel()
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
age               0
workclass         0
fnlwgt            0
education         0
educational-num   0
marital-status    0
occupation        0
relationship      0
race              0
gender            0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

Handling Categorical Data 1 Repalce

print(df[])

In [ ]:
```
print(df['gender'].unique())
```

```
['Male' 'Female']
```

In [ ]:
```
df['gender'] = df['gender'].replace('Male', 1)
df['gender'] = df['gender'].replace('Female', 0)
```

In [ ]:
```
temp_df = pd.DataFrame({
    "Fruit_Name": ['Mango', 'Apple', 'Banana', 'Grapes'],
    "Fruit_Color":['Yellow', 'Red', 'Yellow', 'Green'],
    "Price": [1000,300,50,100]
})

temp_df
```

Out[ ]:

	Fruit_Name	Fruit_Color	Price
0	Mango	Yellow	1000
1	Apple	Red	300
2	Banana	Yellow	50
3	Grapes	Green	100

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

lbl_enc = LabelEncoder()

temp_df['Fruit_Name'] = lbl_enc.fit_transform(temp_df['Fruit_Name']) #randomely assign values to data



temp_df
```

Out[ ]:

	Fruit_Name	Fruit_Color	Price
0	3	Yellow	1000
1	0	Red	300
2	1	Yellow	50
3	2	Green	100

In [ ]:
```
temp_df = pd.get_dummies(temp_df,columns=['Fruit_Color'])#one hot
```

In [ ]:
```
print(df['income'].unique())
```

```
['<=50K' '>50K']
```

In [ ]:
```
df['income'] = df['income'].replace('<=50K', 0)
df['income'] = df['income'].replace('>50K', 1)


df.head()
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	1	0	0	40	United-States	0
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	1	0	0	50	United-States	0
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	1	0	0	40	United-States	1
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	1	7688	0	40	United-States	1
4	18	Private	103497	Some-college	10	Never-married	Prof-specialty	Own-child	White	0	0	0	30	United-States	0

In [ ]:
```
df['marital-status'].unique()
```

Out[ ]:
```
array(['Never-married', 'Married-civ-spouse', 'Widowed', 'Divorced',
       'Separated', 'Married-spouse-absent', 'Married-AF-spouse'],
      dtype=object)
```

In [ ]:
```
df['marital-status'] = df['marital-status'].replace('Never-married', 'Unmarried')
df['marital-status'] = df['marital-status'].replace('Married-civ-spouse', 'Married')
df['marital-status'] = df['marital-status'].replace('Widowed', 'Widowed')
df['marital-status'] = df['marital-status'].replace('Divorced', 'Separated')
df['marital-status'] = df['marital-status'].replace('Separated', 'Separated')
df['marital-status'] = df['marital-status'].replace('Married-spouse-absent', 'Married')
df['marital-status'] = df['marital-status'].replace('Married-AF-spouse', 'Married')
```

In [ ]:
```
df['marital-status'] = lbl_enc.fit_transform(df['marital-status'])

df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	2	Machine-op-inspct	Own-child	Black	1	0	0	40	United-States	0
1	38	Private	89814	HS-grad	9	0	Farming-fishing	Husband	White	1	0	0	50	United-States	0
2	28	Local-gov	336951	Assoc-acdm	12	0	Protective-serv	Husband	White	1	0	0	40	United-States	1
3	44	Private	160323	Some-college	10	0	Machine-op-inspct	Husband	Black	1	7688	0	40	United-States	1
4	18	Private	103497	Some-college	10	2	Prof-specialty	Own-child	White	0	0	0	30	United-States	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	Assoc-acdm	12	0	Tech-support	Wife	White	0	0	0	38	United-States	0
48838	40	Private	154374	HS-grad	9	0	Machine-op-inspct	Husband	White	1	0	0	40	United-States	1
48839	58	Private	151910	HS-grad	9	3	Adm-clerical	Unmarried	White	0	0	0	40	United-States	0
48840	22	Private	201490	HS-grad	9	2	Adm-clerical	Own-child	White	1	0	0	20	United-States	0
48841	52	Self-emp-inc	287927	HS-grad	9	0	Exec-managerial	Wife	White	0	15024	0	40	United-States	1

48790 rows × 15 columns

In [ ]:
```
df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	2	Machine-op-inspct	Own-child	Black	1	0	0	40	United-States	0
1	38	Private	89814	HS-grad	9	0	Farming-fishing	Husband	White	1	0	0	50	United-States	0
2	28	Local-gov	336951	Assoc-acdm	12	0	Protective-serv	Husband	White	1	0	0	40	United-States	1
3	44	Private	160323	Some-college	10	0	Machine-op-inspct	Husband	Black	1	7688	0	40	United-States	1
4	18	Private	103497	Some-college	10	2	Prof-specialty	Own-child	White	0	0	0	30	United-States	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	Assoc-acdm	12	0	Tech-support	Wife	White	0	0	0	38	United-States	0
48838	40	Private	154374	HS-grad	9	0	Machine-op-inspct	Husband	White	1	0	0	40	United-States	1
48839	58	Private	151910	HS-grad	9	3	Adm-clerical	Unmarried	White	0	0	0	40	United-States	0
48840	22	Private	201490	HS-grad	9	2	Adm-clerical	Own-child	White	1	0	0	20	United-States	0
48841	52	Self-emp-inc	287927	HS-grad	9	0	Exec-managerial	Wife	White	0	15024	0	40	United-States	1

48790 rows × 15 columns

In [ ]:
```
df['education'].unique()
```

Out[ ]:
```
array(['11th', 'HS-grad', 'Assoc-acdm', 'Some-college', '10th',
       'Prof-school', '7th-8th', 'Bachelors', 'Masters', 'Doctorate',
       '5th-6th', 'Assoc-voc', '9th', '12th', '1st-4th', 'Preschool'],
      dtype=object)
```

In [ ]:
```
df['education'] = df['education'].replace('Preschool','dropout')
df['education'] = df['education'].replace('10th','dropout')
df['education'] = df['education'].replace('11th','dropout')
df['education'] = df['education'].replace('12th','dropout')
df['education'] = df['education'].replace('1st-4th','dropout')
df['education'] = df['education'].replace('5th-6th','dropout')
df['education'] = df['education'].replace('7th-8th','dropout')
df['education'] = df['education'].replace('9th','dropout')
df['education'] = df['education'].replace('HS-grad','HighGrad')
df['education'] = df['education'].replace('HS-Grad','HighGrad')
df['education'] = df['education'].replace('Some-college','CommunityCollege')
df['education'] = df['education'].replace('Assoc-acdm','CommunityCollege')
df['education'] = df['education'].replace('Assoc-voc','CommunityCollege')
df['education'] = df['education'].replace('Bachelors','Bachelors')
df['education'] = df['education'].replace('Masters','Masters')
```

```
df['education'] = df['education'].replace('Prof-school','Masters')
df['education'] = df['education'].replace('Prof-Doctorate','Doctorate')
```

In [ ]:
```
df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	dropout	7	2	Machine-op-inspct	Own-child	Black	1	0	0	40	United-States	0
1	38	Private	89814	HighGrad	9	0	Farming-fishing	Husband	White	1	0	0	50	United-States	0
2	28	Local-gov	336951	CommunityCollege	12	0	Protective-serv	Husband	White	1	0	0	40	United-States	1
3	44	Private	160323	CommunityCollege	10	0	Machine-op-inspct	Husband	Black	1	7688	0	40	United-States	1
4	18	Private	103497	CommunityCollege	10	2	Prof-specialty	Own-child	White	0	0	0	30	United-States	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	CommunityCollege	12	0	Tech-support	Wife	White	0	0	0	38	United-States	0
48838	40	Private	154374	HighGrad	9	0	Machine-op-inspct	Husband	White	1	0	0	40	United-States	1
48839	58	Private	151910	HighGrad	9	3	Adm-clerical	Unmarried	White	0	0	0	40	United-States	0
48840	22	Private	201490	HighGrad	9	2	Adm-clerical	Own-child	White	1	0	0	20	United-States	0
48841	52	Self-emp-inc	287927	HighGrad	9	0	Exec-managerial	Wife	White	0	15024	0	40	United-States	1

48790 rows × 15 columns

In [ ]:
```
df['education'] = lbl_enc.fit_transform(df['education'])
df['workclass'] = lbl_enc.fit_transform(df['workclass'])
df['occupation'] = lbl_enc.fit_transform(df['occupation'])
df['relationship'] = lbl_enc.fit_transform(df['relationship'])
df['race'] = lbl_enc.fit_transform(df['race'])
df['native-country'] = lbl_enc.fit_transform(df['native-country'])
```

In [ ]:
```
df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	3	226802	5	7	2	6	3	2	1	0	0	40	38	0
1	38	3	89814	3	9	0	4	0	4	1	0	0	50	38	0
2	28	1	336951	1	12	0	10	0	4	1	0	0	40	38	1
3	44	3	160323	1	10	0	6	0	2	1	7688	0	40	38	1
4	18	3	103497	1	10	2	9	3	4	0	0	0	30	38	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	3	257302	1	12	0	12	5	4	0	0	0	38	38	0
48838	40	3	154374	3	9	0	6	0	4	1	0	0	40	38	1
48839	58	3	151910	3	9	3	0	4	4	0	0	0	40	38	0
48840	22	3	201490	3	9	2	0	3	4	1	0	0	20	38	0
48841	52	4	287927	3	9	0	3	5	4	0	15024	0	40	38	1

48790 rows × 15 columns

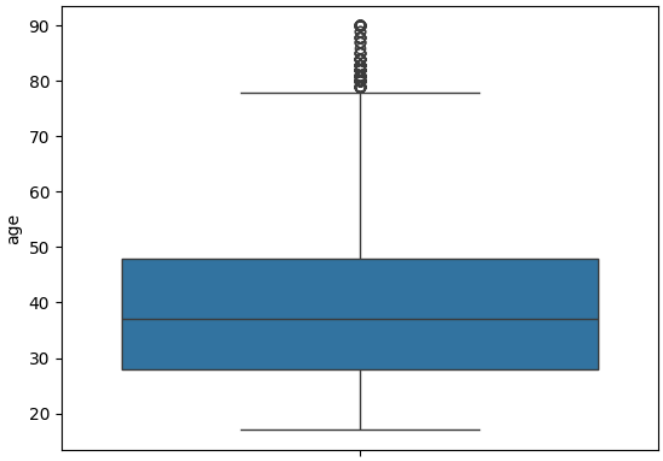
In [ ]:
```
#
```

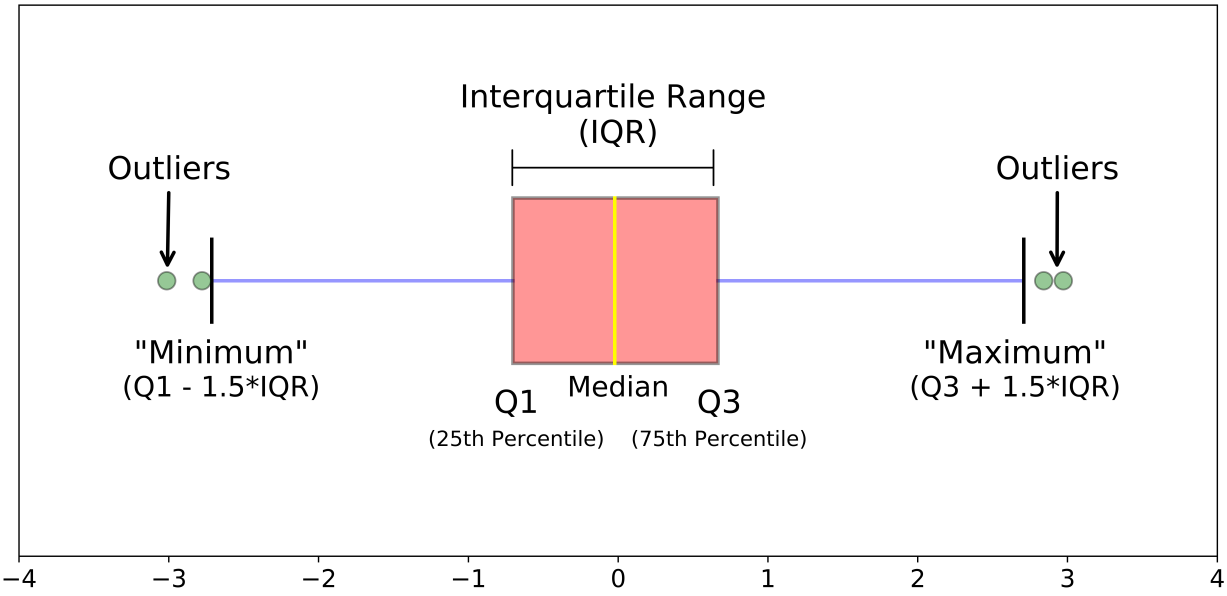
Outlier Detection

1.Boxplot 2.Scatter plot 3.z score 4.inter quartile range

In [ ]:
```
import seaborn as sea
sea.boxplot(df['age'])
```

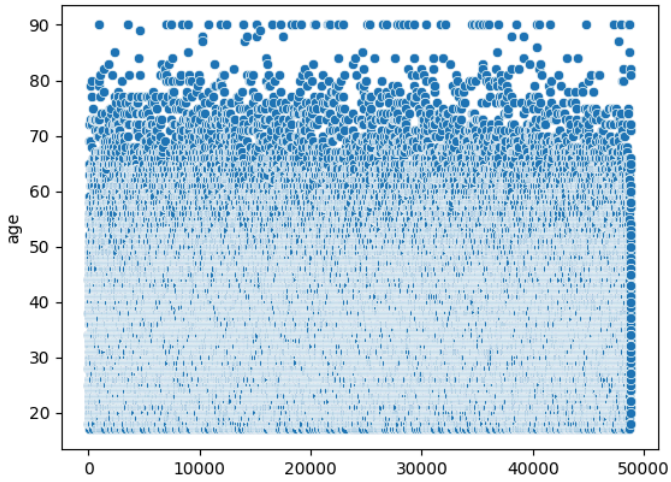
Out[ ]:
```
<Axes: ylabel='age'>
```





In [ ]:
```
sea.scatterplot(df['age'])
```

Out[ ]:
```
<Axes: ylabel='age'>
```



In [ ]:
```
df['age'].unique()
```

Out[ ]:
```
array([25, 38, 28, 44, 18, 34, 29, 63, 24, 55, 65, 36, 26, 58, 48, 43, 20,
       37, 40, 72, 45, 22, 23, 54, 32, 46, 56, 17, 39, 52, 21, 42, 33, 30,
       47, 41, 19, 69, 50, 31, 59, 49, 51, 27, 57, 61, 64, 79, 73, 53, 77,
       80, 62, 35, 68, 66, 75, 60, 67, 71, 70, 90, 81, 74, 78, 82, 83, 85,
       76, 84, 89, 88, 87, 86], dtype=int64)
```

In [ ]:
```
print(np.where(df['age']>78))#prints the index where the age greater than 78
```

```
(array([  193,   234,   898,   925,   950,  1078,  1397,  1833,  2084,
        2289,  2981,  3495,  3667,  4454,  4645,  4657,  6401,  6576,
        6756,  6914,  6958,  6975,  6978,  7159,  7169,  7413,  7418,
        7538,  7546,  7936,  8205,  8312,  8426,  8954,  8981,  9017,
        9037,  9080,  9278,  9768,  9887, 10038, 10198, 10222, 10734,
       11286, 11325, 11407, 11834, 11868, 11878, 11937, 12057, 12226,
       12443, 13022, 13954, 14029, 14259, 14295, 14427, 14564, 14587,
       14736, 15084, 15094, 15404, 15930, 15958, 15998, 16101, 16143,
       16246, 16350, 16498, 16706, 17194, 17316, 17444, 18211, 18578,
       19029, 19166, 19181, 19485, 19612, 19810, 20050, 20236, 20343,
       20382, 20992, 21106, 21542, 21561, 21640, 21676, 22270, 22443,
       22484, 22502, 22709, 22894, 23018, 23751, 23990, 24142, 24445,
       24650, 24700, 24791, 24963, 25075, 25231, 25241, 25737, 26388,
       26474, 26809, 27363, 27502, 27776, 27796, 27994, 28259, 28714,
       28755, 29093, 29238, 29288, 29289, 29557, 29958, 30190, 30366,
       30421, 30865, 30971, 31016, 31163, 31615, 31921, 32151, 32561,
       32782, 33021, 33160, 33867, 34294, 34398, 34529, 34534, 34670,
       34816, 34980, 35087, 35300, 35427, 35435, 35467, 35744, 35750,
       35770, 35943, 36001, 36082, 36503, 36675, 36717, 36736, 36737,
       36862, 37078, 37132, 37205, 37594, 37751, 38062, 38085, 38468,
       38726, 39139, 39142, 39703, 40144, 40271, 40287, 40482, 40524,
       40639, 40804, 41406, 41546, 41640, 42253, 42483, 42971, 44035,
       44415, 44701, 44958, 45184, 45829, 45959, 47261, 47663, 47927,
       48045, 48067, 48086, 48507, 48597, 48688, 48723, 48754],
      dtype=int64),)
```

In [ ]:
```
sorted_df = df.sort_values(by=['age'], ascending=True)

Q1 = np.percentile(sorted_df['age'],25)
Q3 = np.percentile(sorted_df['age'],75)

IQR = Q3 - Q1
print(IQR)
```

```
20.0
```

In [ ]:
```
lower_bound = Q1 - (1.5*IQR)
upper_bound = Q3 + (1.5*IQR)

print("Min: ", lower_bound)
print("Max: ", upper_bound)
```

```
Min:  -2.0
Max:  78.0
```

In [ ]:
```
sorted_df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
22088	17	3	221403	5	8	2	7	3	2	1	0	0	18	38	0
32593	17	3	242773	5	7	2	7	3	4	0	0	0	20	38	0
32598	17	3	133449	5	5	2	7	3	2	1	0	0	26	38	0
14359	17	3	140117	5	7	2	11	3	4	0	0	0	14	38	0
14373	17	3	124661	5	7	2	11	3	4	0	0	0	35	38	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
27793	90	3	87285	3	9	2	7	3	4	0	0	0	24	38	0
48558	90	3	313749	3	9	3	0	4	4	0	0	0	25	38	0
25254	90	3	46786	0	13	0	11	0	4	1	9386	0	15	38	1
10735	90	1	188242	3	9	2	2	3	4	1	11678	0	40	38	1
28277	90	3	40388	0	13	2	3	1	4	1	0	0	55	38	0

48790 rows × 15 columns

In [ ]:
```
outliers = []

for i in df['age']:
    if(i<lower_bound or i > upper_bound):
        outliers.append(i)


print(len(outliers))
print(outliers)
```

```
215
[79, 80, 90, 79, 80, 81, 82, 83, 81, 85, 80, 90, 81, 84, 81, 89, 81, 83, 81, 82, 80, 90, 81, 83, 80, 90, 90, 84, 80, 80, 80, 81, 90, 85, 90, 81, 81,
80, 80, 79, 81, 80, 88, 87, 90, 79, 83, 79, 80, 90, 79, 79, 81, 81, 90, 82, 90, 87, 81, 88, 80, 81, 80, 81, 90, 88, 89, 84, 80, 80, 83, 79, 81, 79, 9
0, 80, 81, 90, 88, 90, 90, 80, 90, 81, 82, 79, 81, 80, 83, 90, 90, 79, 81, 90, 80, 90, 90, 79, 79, 84, 90, 80, 90, 81, 83, 84, 81, 79, 85, 82, 79, 8
0, 90, 90, 90, 84, 80, 90, 90, 79, 84, 90, 79, 90, 90, 90, 82, 81, 90, 84, 79, 81, 82, 81, 80, 90, 80, 84, 82, 79, 90, 84, 90, 83, 79, 81, 80, 79, 8
0, 79, 80, 90, 90, 80, 90, 90, 81, 83, 82, 90, 90, 81, 80, 80, 90, 79, 80, 82, 85, 80, 79, 90, 81, 79, 80, 79, 81, 82, 88, 90, 82, 88, 84, 83, 79, 8
6, 90, 90, 82, 83, 81, 79, 90, 80, 81, 79, 84, 84, 79, 90, 80, 81, 81, 81, 90, 87, 90, 80, 80, 82, 90, 90, 85, 82, 81]
```

Hangling Outliers

1.Removing 2.quartile based flooring and cappin 3.mean/median imputation

In [ ]:
```
#medina way to handling outliers
median = np.median(df['age'])
print(median)
for i in outliers:
    df['age'] = np.where(df['age']==i,37,df['age']) #37 = median
```

```
37.0
```

In [ ]:
```
df['age'].unique()
```

Out[ ]:
```
array([25, 38, 28, 44, 18, 34, 29, 63, 24, 55, 65, 36, 26, 58, 48, 43, 20,
       37, 40, 72, 45, 22, 23, 54, 32, 46, 56, 17, 39, 52, 21, 42, 33, 30,
       47, 41, 19, 69, 50, 31, 59, 49, 51, 27, 57, 61, 64, 73, 53, 77, 62,
       35, 68, 66, 75, 60, 67, 71, 70, 74, 78, 76], dtype=int64)
```

In [ ]:
```
df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	3	226802	5	7	2	6	3	2	1	0	0	40	38	0
1	38	3	89814	3	9	0	4	0	4	1	0	0	50	38	0
2	28	1	336951	1	12	0	10	0	4	1	0	0	40	38	1
3	44	3	160323	1	10	0	6	0	2	1	7688	0	40	38	1
4	18	3	103497	1	10	2	9	3	4	0	0	0	30	38	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	3	257302	1	12	0	12	5	4	0	0	0	38	38	0
48838	40	3	154374	3	9	0	6	0	4	1	0	0	40	38	1
48839	58	3	151910	3	9	3	0	4	4	0	0	0	40	38	0
48840	22	3	201490	3	9	2	0	3	4	1	0	0	20	38	0
48841	52	4	287927	3	9	0	3	5	4	0	15024	0	40	38	1

48790 rows × 15 columns

In [ ]:
```
df['age'].sort_values().unique()
```

Out[ ]:
```
array([17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78], dtype=int64)
```

Data Sampling

In [ ]:
```
df['income'].value_counts()
```

Out[ ]:
```
income
0    37109
1    11681
Name: count, dtype: int64
```

In [ ]:
```
#random sampling

lt_fiftyk = df[df['income']==0]
gt_fiftyk = df[df['income']==1]

print("<=50k: ",lt_fiftyk.shape)
print(">50k: ",gt_fiftyk.shape)
```

```
<=50k:  (37109, 15)
>50k:  (11681, 15)
```

In [ ]:
```
no_sample = lt_fiftyk.sample(n=11681)
```

In [ ]:
```
no_sample.shape
```

Out[ ]:
```
(11681, 15)
```

In [ ]:
```
sampled_df = pd.concat([no_sample,gt_fiftyk],axis=0)
```

In [ ]:
```
sampled_df.shape
```

Out[ ]:
```
(23362, 15)
```

In [ ]:
```
sampled_df
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
12503	67	3	221281	3	9	1	0	1	4	0	0	0	15	38	0
10300	17	3	219199	5	6	2	7	3	2	1	0	0	15	38	0
35430	22	3	138513	1	10	2	2	3	4	1	0	0	40	38	0
20835	18	3	353358	1	10	2	7	3	4	1	0	0	16	38	0
39659	53	1	204397	3	9	0	3	0	4	1	0	0	60	38	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48820	71	3	287372	2	16	0	9	0	4	1	0	0	10	38	1
48826	39	1	111499	1	12	0	0	5	4	0	0	0	20	38	1
48835	53	3	321865	4	14	0	3	0	4	1	0	0	40	38	1
48838	40	3	154374	3	9	0	6	0	4	1	0	0	40	38	1
48841	52	4	287927	3	9	0	3	5	4	0	15024	0	40	38	1

23362 rows × 15 columns

In [ ]:
```
sampled_df['income'].value_counts()
```

Out[ ]:
```
income
0    11681
1    11681
Name: count, dtype: int64
```

In [ ]:
```
X = df.drop('income', axis=1)
y = df['income']
```

In [ ]:
```
print("Shape of X: ", X.shape)
print("Shape of y: ", y.shape)
```

```
Shape of X:  (48790, 14)
Shape of y:  (48790,)
```

In [ ]:
```
df.corr()
```

Out[ ]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	inco
age	1.000000	0.044471	-0.073595	0.063870	0.036319	-0.345357	-0.002090	-0.265529	0.027914	0.089179	0.077915	0.056658	0.087974	-0.002915	0.238
workclass	0.044471	1.000000	-0.026516	0.011353	0.007331	-0.054750	0.009878	-0.056085	0.053939	0.066675	0.031554	0.004160	0.042887	-0.004872	-0.000
fnlwgt	-0.073595	-0.026516	1.000000	0.019093	-0.038727	0.021973	-0.002391	0.009017	-0.027165	0.027879	-0.003715	-0.004378	-0.013521	-0.058299	-0.006
education	0.063870	0.011353	0.019093	1.000000	-0.605655	0.003738	0.006641	0.021186	-0.020251	0.033225	-0.006344	-0.024376	-0.060474	-0.081753	-0.134
educational-num	0.036319	0.007331	-0.038727	-0.605655	1.000000	-0.077254	0.072688	-0.090697	0.029331	0.009364	0.125219	0.080986	0.143915	0.089359	0.332
marital-status	-0.345357	-0.054750	0.021973	0.003738	-0.077254	1.000000	0.003616	0.439649	-0.075049	-0.370367	-0.077925	-0.067813	-0.244835	0.020400	-0.406
occupation	-0.002090	0.009878	-0.002391	0.006641	0.072688	0.003616	1.000000	-0.035054	-0.005158	0.042773	0.014498	0.011048	-0.015454	-0.001643	0.032
relationship	-0.265529	-0.056085	0.009017	0.021186	-0.090697	0.439649	-0.035054	1.000000	-0.116985	-0.579955	-0.056543	-0.057243	-0.250319	-0.007092	-0.253
race	0.027914	0.053939	-0.027165	-0.020251	0.029331	-0.075049	-0.005158	-0.116985	1.000000	0.086959	0.011610	0.018640	0.039759	0.117740	0.070
gender	0.089179	0.066675	0.027879	0.033225	0.009364	-0.370367	0.042773	-0.579955	0.086959	1.000000	0.047127	0.045517	0.228529	-0.002544	0.214
capital-gain	0.077915	0.031554	-0.003715	-0.006344	0.125219	-0.077925	0.014498	-0.056543	0.011610	0.047127	1.000000	-0.031475	0.082152	0.007884	0.223
capital-loss	0.056658	0.004160	-0.004378	-0.024376	0.080986	-0.067813	0.011048	-0.057243	0.018640	0.045517	-0.031475	1.000000	0.054431	0.006466	0.147
hours-per-week	0.087974	0.042887	-0.013521	-0.060474	0.143915	-0.244835	-0.015454	-0.250319	0.039759	0.228529	0.082152	0.054431	1.000000	0.006668	0.227
native-country	-0.002915	-0.004872	-0.058299	-0.081753	0.089359	0.020400	-0.001643	-0.007092	0.117740	-0.002544	0.007884	0.006466	0.006668	1.000000	0.020
income	0.238085	-0.000508	-0.006309	-0.134587	0.332802	-0.406910	0.032533	-0.253175	0.070970	0.214639	0.223047	0.147542	0.227664	0.020161	1.000

In [ ]:
```
#feature selection - increases computation efficiency
from sklearn.feature_selection import mutual_info_classif

mutual_info = mutual_info_classif(X,y)

mutual_info
```

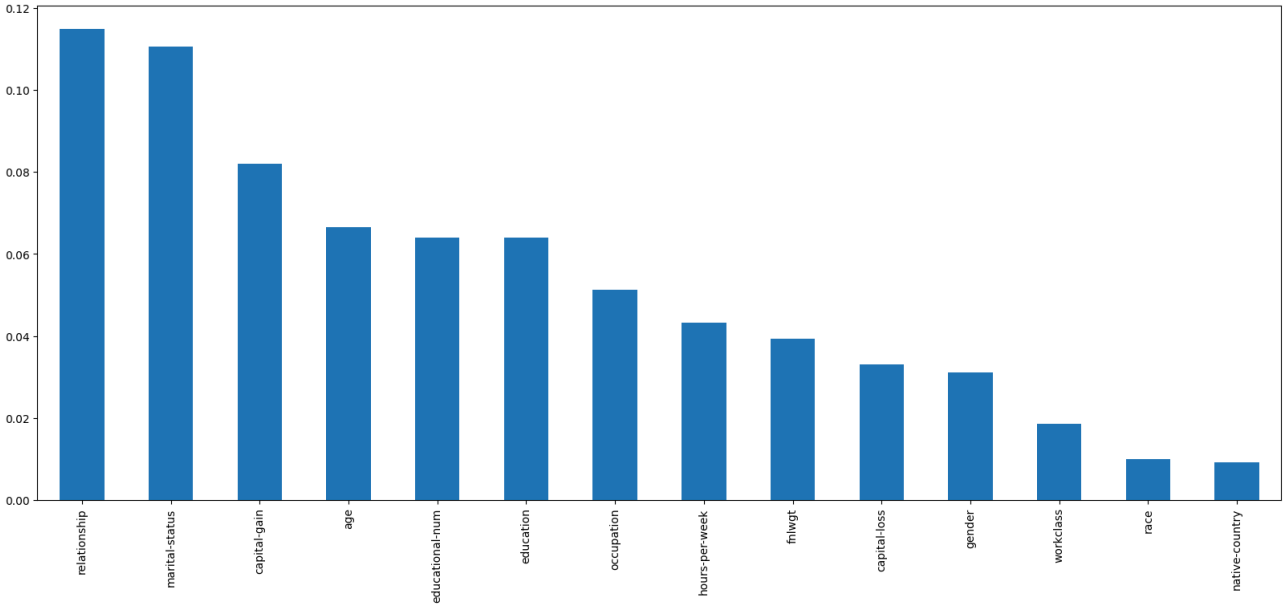
Out[ ]:
```
array([0.06651747, 0.01863082, 0.03931678, 0.06391469, 0.0640368 ,
       0.1106235 , 0.05128719, 0.1148209 , 0.00996512, 0.03117667,
       0.08205614, 0.03303608, 0.04324361, 0.00924937])
```

In [ ]:
```
mutual_info = pd.Series(mutual_info)
mutual_info.index = X.columns
mutual_info.sort_values(ascending=False)
```

Out[ ]:
```
relationship       0.114821
marital-status     0.110624
capital-gain       0.082056
age                0.066517
educational-num    0.064037
education          0.063915
occupation         0.051287
hours-per-week     0.043244
fnlwgt             0.039317
capital-loss       0.033036
gender             0.031177
workclass          0.018631
race               0.009965
native-country     0.009249
dtype: float64
```

In [ ]:
```
mutual_info.sort_values(ascending=False).plot.bar(figsize=(20,8))
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
X = df.drop(['native-country', 'race', 'workclass', 'gender', 'capital-loss','income'],axis=1)

X
```

Out[ ]:

	age	fnlwgt	education	educational-num	marital-status	occupation	relationship	capital-gain	hours-per-week
0	25	226802	5	7	2	6	3	0	40
1	38	89814	3	9	0	4	0	0	50
2	28	336951	1	12	0	10	0	0	40
3	44	160323	1	10	0	6	0	7688	40
4	18	103497	1	10	2	9	3	0	30
...	...	...	...	...	...	...	...	...	...
48837	27	257302	1	12	0	12	5	0	38
48838	40	154374	3	9	0	6	0	0	40
48839	58	151910	3	9	3	0	4	0	40
48840	22	201490	3	9	2	0	3	0	20
48841	52	287927	3	9	0	3	5	15024	40

48790 rows × 9 columns

In [ ]:
```
y
```

Out[ ]:
```
0        0
1        0
2        1
3        1
4        0
        ..
48837    0
48838    1
48839    0
48840    0
48841    1
Name: income, Length: 48790, dtype: int64
```

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=42, shuffle=True)
```

In [ ]:
```
print('X training shape: ', X_train.shape)
print('X test shape: ', X_test.shape)
print('y training shape: ', y_train.shape)
print('y test shape: ', y_test.shape)
```

```
X training shape:  (34153, 9)
X test shape:  (14637, 9)
y training shape:  (34153,)
y test shape:  (14637,)
```

In [ ]:
```
#now we scale the dataset, we try to bring the mean similar to each other - DATA SCALING, a single value cant overpower the smaller values


#standard scaler

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)# scaling of mean of the 80% training set
X_test_std = scaler.transform(X_test) # scaling of mean by on old mean as we only have 20% of records for testing  (bigger chunks of data with big ran
```

In [ ]:
```
X_train_std.shape
```

Out[ ]:
```
(34153, 9)
```

In [ ]:
```
X_test_std.shape
```

Out[ ]:
```
(14637, 9)
```

In [ ]:
```
#now model is ready to train

#number of perceptron depend, trail and error

# import tensorflow as tf
# from tensorflow.keras.layers import Dense
```

In [ ]:
```
#MLP classifier

from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(8,), activation='relu', batch_size=32, verbose=True, max_iter=10, solver='sgd')
```

class sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(100,), activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000)

In [ ]:
```
mlp
```

Out[ ]:
```
MLPClassifier(batch_size=32, hidden_layer_sizes=(8,), max_iter=10, solver='sgd',
              verbose=True)
```

In [ ]:
```
mlp.fit(X_train_std, y_train)
```

```
Iteration 1, loss = 0.45846687
Iteration 2, loss = 0.37329158
Iteration 3, loss = 0.35353009
Iteration 4, loss = 0.34715745
Iteration 5, loss = 0.34412023
Iteration 6, loss = 0.34199146
Iteration 7, loss = 0.34039237
Iteration 8, loss = 0.33932211
Iteration 9, loss = 0.33829771
Iteration 10, loss = 0.33757126
c:\Users\nilesh\anaconda3\envs\mllab\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (10) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[ ]:
```
MLPClassifier(batch_size=32, hidden_layer_sizes=(8,), max_iter=10, solver='sgd',
              verbose=True)
```

In [ ]:
```
mlp.score(X_train_std, y_train)
```

Out[ ]:
```
0.8416830146692823
```

In [ ]:
```
mlp.score(X_test_std, y_test)
```

Out[ ]:
```
0.8335041333606613
```

In [ ]:
```
y_pred_skln = mlp.predict(X_test_std)
```

In [ ]:
```
print(y_pred_skln)
```

```
[0 0 1 ... 0 1 0]
```

In [ ]:
```
#confusion matrix

from sklearn.metrics import confusion_matrix, classification_report
```

In [ ]:
```
cm =  confusion_matrix(y_test, y_pred_skln)

print(cm)
```

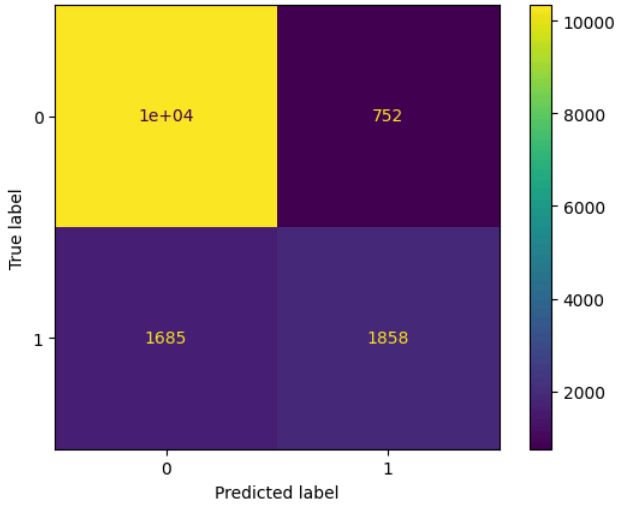
```
[[10342   752]
 [ 1685  1858]]
```

In [ ]:
```
cr = classification_report(y_test, y_pred_skln)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.86      0.93      0.89     11094
           1       0.71      0.52      0.60      3543

    accuracy                           0.83     14637
   macro avg       0.79      0.73      0.75     14637
weighted avg       0.82      0.83      0.82     14637
```

In [ ]:
```
from sklearn.metrics import ConfusionMatrixDisplay

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x234afb1d6c0>
```



In [ ]:
```
from sklearn import metrics

fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_skln)

roc_auc = metrics.auc(fpr, tpr)

display = metrics.RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc, estimator_name="temp")

display.plot()
```

Out[ ]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x234afb5ae30>
```