FODS Assignment 3

Aim:- Perform the following operations using Python on your dataset

1. Data Cleaning
2. Data Transformation
3. Error Correcting

Data cleaning:- Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

Data Transformation:- Data transformation is the process of converting data from one format to another, typically from the format of a source system into the required format of a destination system. Data transformation is a component of most data integration and data management tasks, such as data wrangling and data warehousing.

Error Correcting:- Error detection and correction code plays an important role in the transmission of data from one source to another. The noise also gets added into the data when it transmits from one system to another, which causes errors in the received binary data at other systems. The bits of the data may change(either 0 to 1 or 1 to 0) during transmission.

It is impossible to avoid the interference of noise, but it is possible to get back the original data. For this purpose, we first need to detect either an error **z** is present or not using error detection codes. If the error is present in the code, then we will correct it with the help of error correction codes.

import numpy as np

import pandas as pd

covid19 = pd.read\_csv('C:/Users/Lenovo/Desktop/Covid19\_India.csv') covid19

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \ | Date |  | State TotalSamples | Negative |
| 0 | 2020-04-17 Andaman | and Nicobar | Islands 1403.0 | 1210 |
| 1 | 2020-04-24 Andaman | and Nicobar | Islands 2679.0 | NaN |
| 2 | 2020-04-27 Andaman | and Nicobar | Islands 2848.0 | NaN |
| 3 | 2020-05-01 Andaman | and Nicobar | Islands 3754.0 | NaN |
| 4 | 2020-05-16 Andaman | and Nicobar | Islands 6677.0 | NaN |
| ... | ... |  | ... ... | ... |
| 16331 | 2021-08-06 | West | Bengal 15999961.0 | NaN |
| 16332 | 2021-08-07 | West | Bengal 16045662.0 | NaN |
| 16333 | 2021-08-08 | West | Bengal 16092192.0 | NaN |
| 16334 | 2021-08-09 | West | Bengal 16122345.0 | NaN |
| 16335 | 2021-08-10 | West | Bengal 16162814.0 | NaN |
| 0 | Positive  12.0 |  | | |
| 1 | 27.0 |
| 2 | 33.0 |
| 3 | 33.0 |
| 4  ... 16331 | 33.0  ...  NaN |
| 16332 | NaN |
| 16333 | NaN |
| 16334 | NaN |
| 16335 | NaN |
| [16336 | rows x 5 columns] |

covid19.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 16336 entries, 0 to 16335 Data columns (total 5 columns):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # |  | Column | Non-Null Count |  | Dtype |
| 0 |  | Date | 16336 non-null |  | object |
| 1 |  | State | 16336 non-null |  | object |
| 2 |  | TotalSamples | 16336 non-null |  | float64 |
| 3 |  | Negative | 6969 non-null |  | object |
| 4 |  | Positive | 5662 non-null |  | float64 |

dtypes: float64(2), object(3) memory usage: 638.2+ KB

*#Import library related to Label Encoder*

from sklearn.preprocessing import LabelEncoder

*#Creating an instance of LabelEncoder*

labelencoder = LabelEncoder()

*#Assigning numerical values and storing it in another column called "State\_N"*

covid19["State\_N"] = labelencoder.fit\_transform(covid19["State"])

*#Display of dataframe*

covid19

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Date | | State | | | | TotalSamples | Negative |
| \ 0 | 2020-04-17 | Andaman | and | Nicobar | Islands | 1403.0 | 1210 |
| 1 | 2020-04-24 | Andaman | and | Nicobar | Islands | 2679.0 | NaN |
| 2 | 2020-04-27 | Andaman | and | Nicobar | Islands | 2848.0 | NaN |
| 3 | 2020-05-01 | Andaman | and | Nicobar | Islands | 3754.0 | NaN |
| 4  ... 16331 | 2020-05-16  ... 2021-08-06 | Andaman | and | Nicobar  West | Islands  ...  Bengal | 6677.0  ... 15999961.0 | NaN  ...  NaN |
| 16332 | 2021-08-07 |  |  | West | Bengal | 16045662.0 | NaN |
| 16333 | 2021-08-08 |  |  | West | Bengal | 16092192.0 | NaN |
| 16334 | 2021-08-09 |  |  | West | Bengal | 16122345.0 | NaN |
| 16335 | 2021-08-10 |  |  | West | Bengal | 16162814.0 | NaN |
|  | Positive | State\_N |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| 0 | 12.0 | 0 |
| 1 | 27.0 | 0 |
| 2 | 33.0 | 0 |
| 3 | 33.0 | 0 |
| 4 | 33.0 | 0 |
| ... | ... | ... |
| 16331 | NaN | 35 |
| 16332 | NaN | 35 |
| 16333 | NaN | 35 |
| 16334 | NaN | 35 |
| 16335 | NaN | 35 |

[16336 rows x 6 columns]

covid19["State"] = covid19["State"].astype('category') covid19.dtypes

Date object

State category

TotalSamples float64

Negative object

Positive float64

State\_N int32

dtype: object

covid19["State"] = covid19["State"].cat.codes covid19

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Date | State | TotalSamples | Negative | Positive | State\_N |
| 0 | 2020-04-17 | 0 | 1403.0 | 1210 | 12.0 | 0 |
| 1 | 2020-04-24 | 0 | 2679.0 | NaN | 27.0 | 0 |
| 2 | 2020-04-27 | 0 | 2848.0 | NaN | 33.0 | 0 |
| 3 | 2020-05-01 | 0 | 3754.0 | NaN | 33.0 | 0 |
| 4 | 2020-05-16 | 0 | 6677.0 | NaN | 33.0 | 0 |
| ... | ... | ... | ... | ... | ... | ... |
| 16331 | 2021-08-06 | 35 | 15999961.0 | NaN | NaN | 35 |
| 16332 | 2021-08-07 | 35 | 16045662.0 | NaN | NaN | 35 |
| 16333 | 2021-08-08 | 35 | 16092192.0 | NaN | NaN | 35 |
| 16334 | 2021-08-09 | 35 | 16122345.0 | NaN | NaN | 35 |
| 16335 | 2021-08-10 | 35 | 16162814.0 | NaN | NaN | 35 |

[16336 rows x 6 columns]

*#Import library related to One Hot Encoder*

from sklearn.preprocessing import OneHotEncoder

*#Create the OneHotEncoder object*

onehotencoder = OneHotEncoder(sparse = False, handle\_unknown = 'error', drop = 'first')

*#Perform OneHotEncoding to a column called "State" using onehotencoder*

*object fit and transform*

onehotencoder\_df = pd.DataFrame(onehotencoder.fit\_transform(covid19[["State"]])) onehotencoder\_df

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ... 25 | 26 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... ... ... | ... |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 28 | 29 | 30 | 31 | 32 | 33 | 34 |  | | | |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |

27 \

0

0.0

1

0.0

2

0.0

3

0.0

4

0.0

...

... 16331

0.0

16332

0.0

16333

0.0

16334

0.0

16335

0.0

0

1

2

3

4

... 16331

16332

16333

16334

16335

[16336 rows x 35 columns]

covid19 = covid19.join(onehotencoder\_df) covid19

Date State TotalSamples Negative Positive State\_N

0 1 \

0 2020-04-17 0 1403.0 1210 12.0 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0 | 0.0 | |  | | | | | | | | | |
| 1 | 2020-04-24 | | 0 | 2679.0 | | | NaN | 27.0 | | 0 | |  |
| 0.0 | 0.0 | |  |  | | |  |  | |  | |  |
| 2 | 2020-04-27 | | 0 | 2848.0 | | | NaN | 33.0 | | 0 | |  |
| 0.0 | 0.0 | |  |  | | |  |  | |  | |  |
| 3 | 2020-05-01 | | 0 | 3754.0 | | | NaN | 33.0 | | 0 | |  |
| 0.0 | 0.0 | |  |  | | |  |  | |  | |  |
| 4 | 2020-05-16 | | 0 | 6677.0 | | | NaN | 33.0 | | 0 | |  |
| 0.0  ... | 0.0  ... | | ... | ... | | | ... | ... | | ... | | .. |
| . ...  16331 2021-08-06 | | | 35 | 15999961.0 | | | NaN | NaN | | 35 | | |
| 0.0 0.0 | | |  |  | | |  |  | |  | | |
| 16332 2021-08-07 | | | 35 | 16045662.0 | | | NaN | NaN | | 35 | | |
| 0.0 0.0 | | |  |  | | |  |  | |  | | |
| 16333 2021-08-08 | | | 35 | 16092192.0 | | | NaN | NaN | | 35 | | |
| 0.0 0.0 | | |  |  | | |  |  | |  | | |
| 16334 2021-08-09 | | | 35 | 16122345.0 | | | NaN | NaN | | 35 | | |
| 0.0 0.0 | | |  |  | | |  |  | |  | | |
| 16335 2021-08-10 | | | 35 | 16162814.0 | | | NaN | NaN | | 35 | | |
| 0.0 | 0.0 |  | |  |  |  |  |  |  |  |  |  |
|  | 2 | 3 ... 25 | | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 0 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ... | ... | ... ... ... | | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 16331 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 16332 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 16333 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 16334 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 16335 | 0.0 | 0.0 ... 0.0 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |

[16336 rows x 41 columns]

FODS Assignment 3 (part B)

*# Libraries for Exploratory Data Analysis*

import os

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

%matplotlib inline

df = pd.read\_csv('C:/Users/Lenovo/Desktop/heart.csv') df.head(3)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| age slope | sex  \ | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak |
| 0 63 1 | | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 |
| 1 37 1 | | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 |
| 2 41 0 | | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 |

0

0

2

ca thal target 0 0 1 1

1 0 2 1

2 0 2 1

df.shape (303, 14)

df.columns

Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',

'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'], dtype='object')

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 303 entries, 0 to 302 Data columns (total 14 columns):

# Column Non-Null Count Dtype

1. age 303 non-null int64
2. sex 303 non-null int64
3. cp 303 non-null int64
4. trestbps 303 non-null int64
5. chol 303 non-null int64
6. fbs 303 non-null int64
7. restecg 303 non-null int64
8. thalach 303 non-null int64
9. exang 303 non-null int64

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | oldpeak | 303 non-null | float64 |
| 10 | slope | 303 non-null | int64 |
| 11 | ca | 303 non-null | int64 |
| 12 | thal | 303 non-null | int64 |
| 13 | target | 303 non-null | int64 |

dtypes: float64(1), int64(13) memory usage: 33.3 KB

there are no nulls

## Check data type

*# to know the type of variable*

df.nunique()

age 41

sex 2

cp 4

trestbps 49

chol 152

fbs 2

restecg 3

thalach 91

exang 2

oldpeak 40

slope 3

ca 5

thal 4

target 2

dtype: int64 df.dtypes

age int64

sex int64

cp int64

trestbps int64

chol int64

fbs int64

restecg int64

thalach int64

exang int64 oldpeak float64 slope int64

ca int64

thal int64

target int64 dtype: object

df['ca'].unique()

array([0, 2, 1, 3, 4], dtype=int64)

*# to count the number in of each category decending order*

df.ca.value\_counts()

|  |  |
| --- | --- |
| 0 | 175 |
| 1 | 65 |
| 2 | 38 |
| 3 | 20 |
| 4 | 5 |

Name: ca, dtype: int64

*# to check missing values*

df.isnull().sum()

age 0

sex 0

cp 0

trestbps 0

chol 0

fbs 0

restecg 0

thalach 0

exang 0

oldpeak 0

slope 0

ca 0

thal 0

target 0

dtype: int64

*# change the labelling for better interpretation/ visualization understanding*

df['target'] = df.target.replace({1: "Disease", 0: "No\_disease"}) df['sex'] = df.sex.replace({1: "Male", 0: "Female"})

df['cp'] = df.cp.replace({1: "typical\_angina",

2: "atypical\_angina", 3:"non-anginal pain", 4: "asymtomatic"})

df['exang'] = df.exang.replace({1: "Yes", 0: "No"}) df['slope'] = df.cp.replace({1: "upsloping",

2: "flat", 3:"downsloping"})

df['thal'] = df.thal.replace({1: "fixed\_defect", 2: "reversable\_defect", 3:"normal"})

*# to know the basic stats*

df.describe()

age trestbps chol fbs restecg

thalach \

count 303.000000 303.000000 303.000000 303.000000 303.000000

303.000000

mean 54.366337 131.623762 246.264026 0.148515 0.528053

149.646865

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| std 9.082101 | 17.538143 | 51.830751 | 0.356198 | 0.525860 |
| 22.905161 |  |  |  |  |
| min 29.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000000 |

71.000000

25% 47.500000 120.000000 211.000000 0.000000 0.000000

133.500000

50% 55.000000 130.000000 240.000000 0.000000 1.000000

153.000000

75% 61.000000 140.000000 274.500000 0.000000 1.000000

166.000000

max 77.000000 200.000000 564.000000 1.000000 2.000000

202.000000

|  |  |  |
| --- | --- | --- |
|  | oldpeak | ca |
| count | 303.000000 | 303.000000 |
| mean | 1.039604 | 0.729373 |
| std | 1.161075 | 1.022606 |
| min | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 |
| 50% | 0.800000 | 0.000000 |
| 75% | 1.600000 | 1.000000 |
| max | 6.200000 | 4.000000 |

## EDA on Heart Disease Dataset

df.columns

Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',

'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'], dtype='object')

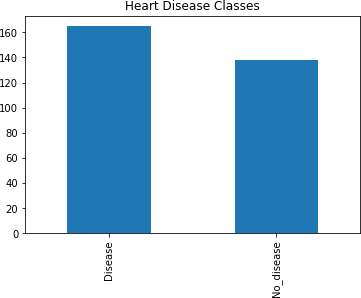
print(df.target.value\_counts()) df['target'].value\_counts().plot(kind='bar').set\_title('Heart Disease Classes')

Disease 165

No\_disease 138

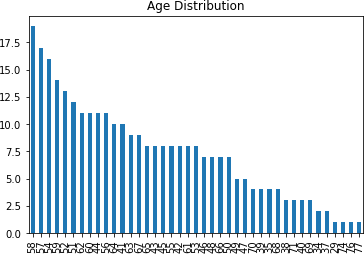
Name: target, dtype: int64

Text(0.5, 1.0, 'Heart Disease Classes')



*# print(df.age.value\_counts())* df['age'].value\_counts().plot(kind='bar').set\_title('Age Distribution')

Text(0.5, 1.0, 'Age Distribution')



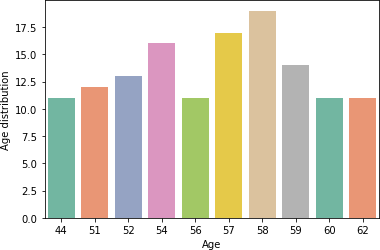
*# Analyze distribution in age in range 10* print(df.age.value\_counts()[:10]) sns.barplot(x=df.age.value\_counts()[:10].index,

y=df.age.value\_counts()[:10].values, palette='Set2')

plt.xlabel('Age') plt.ylabel('Age distribution')

|  |  |
| --- | --- |
| 58 | 19 |
| 57 | 17 |
| 54 | 16 |
| 59 | 14 |
| 52 | 13 |
| 51 | 12 |
| 62 | 11 |
| 60 | 11 |
| 44 | 11 |
| 56 | 11 |
| Name: | age, dtype: int64 |

Text(0, 0.5, 'Age distribution')



*# to know the youngest or oldest in age*

print(min(df.age)) print(max(df.age)) print(df.age.mean())

29

77

54.366336633663366

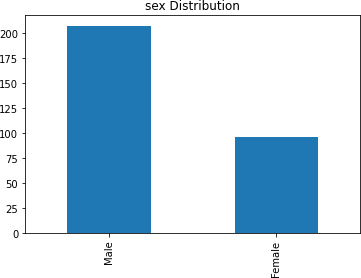
print(df.sex.value\_counts()) df['sex'].value\_counts().plot(kind='bar').set\_title('sex Distribution')

Male 207

Female 96

Name: sex, dtype: int64

Text(0.5, 1.0, 'sex Distribution')



print(df.cp.value\_counts()) df['cp'].value\_counts().plot(kind='bar').set\_title('Chest Pain Distribution')

0 143

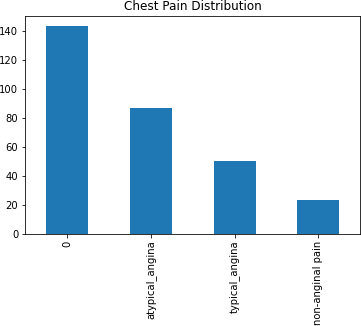
atypical\_angina 87

typical\_angina 50

non-anginal pain 23

Name: cp, dtype: int64

Text(0.5, 1.0, 'Chest Pain Distribution')



print(df.restecg.value\_counts()) df['restecg'].value\_counts().plot(kind='bar').set\_title('Resting ECG Distribution')

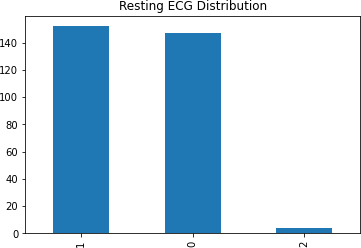
1 152

0 147

2 4

Name: restecg, dtype: int64

Text(0.5, 1.0, 'Resting ECG Distribution')



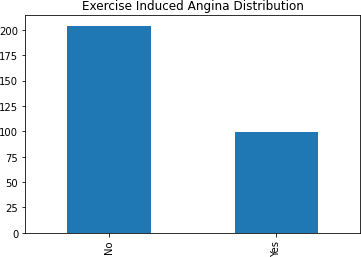
print(df.exang.value\_counts()) df['exang'].value\_counts().plot(kind='bar').set\_title('Exercise Induced Angina Distribution')

No 204

Yes 99

Name: exang, dtype: int64

Text(0.5, 1.0, 'Exercise Induced Angina Distribution')

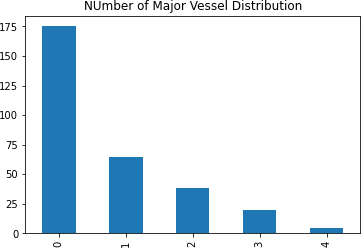


print(df.ca.value\_counts()) df['ca'].value\_counts().plot(kind='bar').set\_title('NUmber of Major Vessel Distribution')

|  |  |
| --- | --- |
| 0 | 175 |
| 1 | 65 |
| 2 | 38 |
| 3 | 20 |
| 4 | 5 |

Name: ca, dtype: int64

Text(0.5, 1.0, 'NUmber of Major Vessel Distribution')



print(df.thal.value\_counts()) df['thal'].value\_counts().plot(kind='bar').set\_title('thal Distribution')

reversable\_defect 166

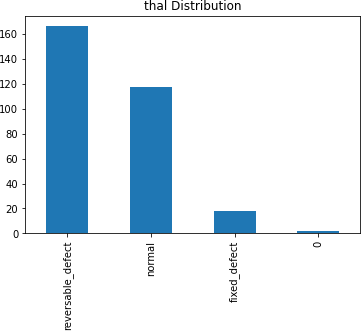
normal 117

fixed\_defect 18

0 2

Name: thal, dtype: int64

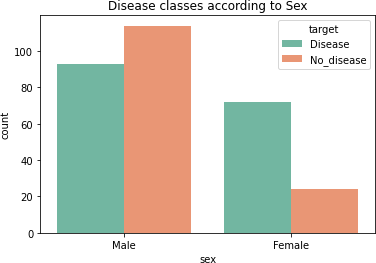
Text(0.5, 1.0, 'thal Distribution')



# Visualize categorical data distribution

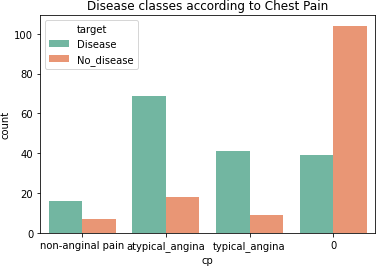
sns.countplot(x='sex', hue='target', data=df, palette='Set2').set\_title('Disease classes according to Sex')

Text(0.5, 1.0, 'Disease classes according to Sex')



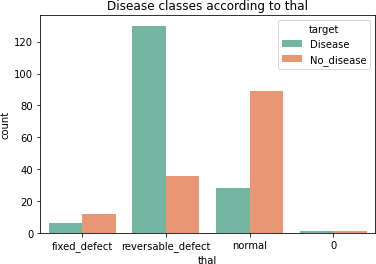
sns.countplot(x='cp', hue='target', data=df, palette='Set2').set\_title('Disease classes according to Chest Pain')

Text(0.5, 1.0, 'Disease classes according to Chest Pain')



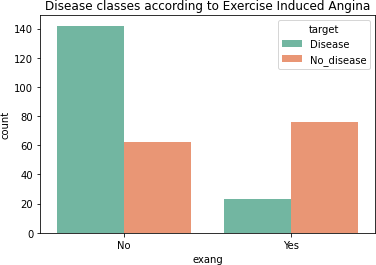
sns.countplot(x='thal', hue='target', data=df, palette='Set2').set\_title('Disease classes according to thal')

Text(0.5, 1.0, 'Disease classes according to thal')



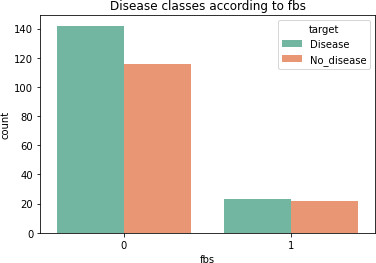
sns.countplot(x='exang', hue='target', data=df, palette='Set2').set\_title('Disease classes according to Exercise Induced Angina')

Text(0.5, 1.0, 'Disease classes according to Exercise Induced Angina')



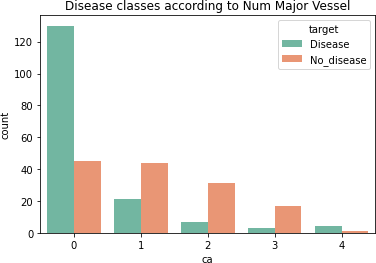
sns.countplot(x='fbs', hue='target', data=df, palette='Set2').set\_title('Disease classes according to fbs')

Text(0.5, 1.0, 'Disease classes according to fbs')



sns.countplot(x='ca', hue='target', data=df, palette='Set2').set\_title('Disease classes according to Num Major Vessel')

Text(0.5, 1.0, 'Disease classes according to Num Major Vessel')



# Visualize all together

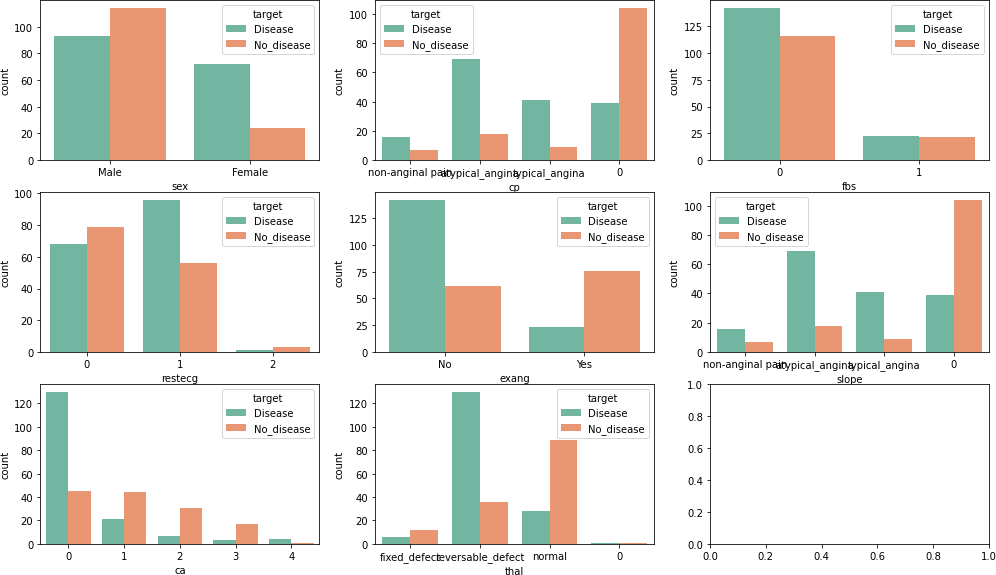
*# for plotting, group categorical features in cat\_feat # to create dist in 8 feature, 9th is the target,*

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(17,10)) cat\_feat = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

**for** idx, feature **in** enumerate(cat\_feat): ax = axes[int(idx/3), idx%3]

**if** feature != 'target':

sns.countplot(x=feature, hue='target', data=df, ax=ax, palette='Set2')



*# Another way of visualizing: Pie charts for thalassemia Having heart disease*

labels= 'Normal', 'Fixed defect', 'Reversible defect' sizes=[6, 130, 28]

colors=['pink', 'orange', 'purple']

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%', startangle=140)

plt.axis('equal')

plt.title('Thalassemia with heart disease') plt.show()

*# Not having heart disease*

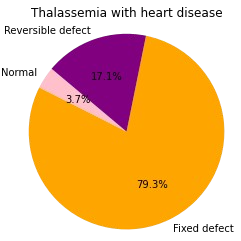
labels= 'Normal', 'Fixed defect', 'Reversible defect' sizes=[12, 36, 89]

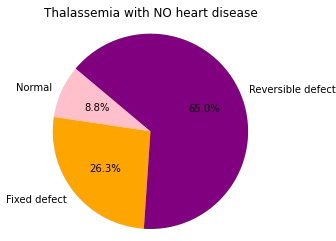
colors=['pink', 'orange', 'purple']

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%', startangle=140)

plt.axis('equal')

plt.title('Thalassemia with NO heart disease') plt.show()





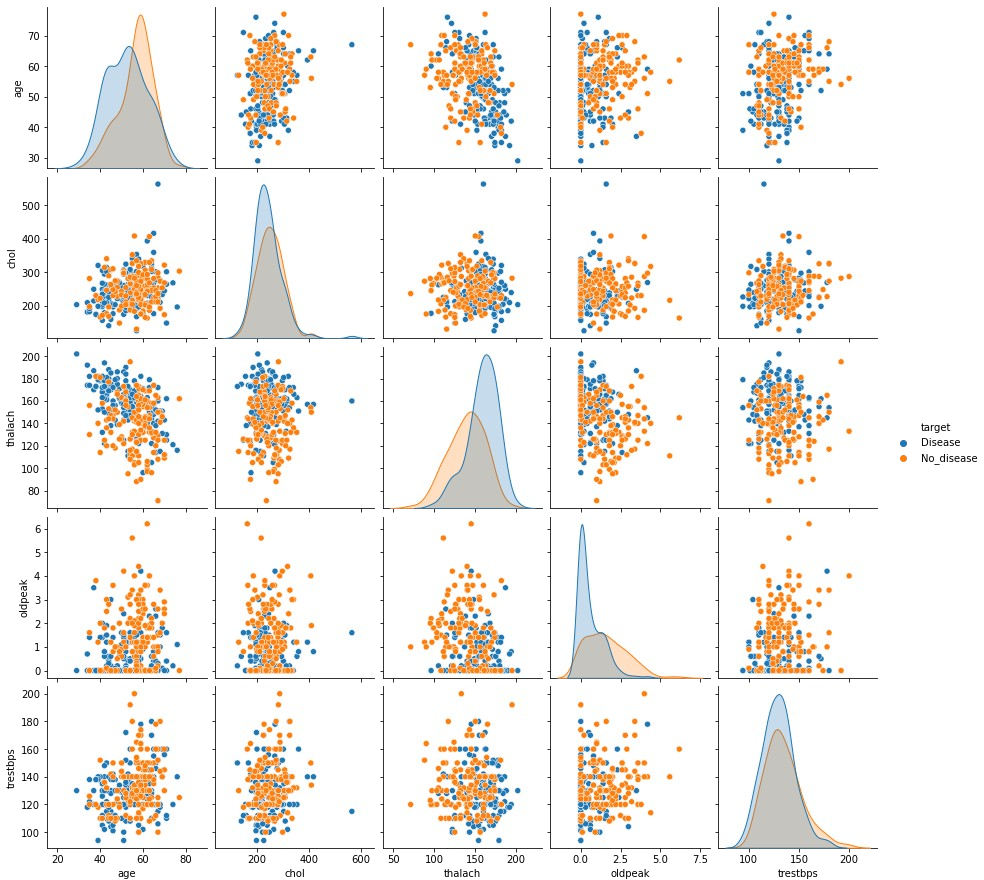
## Visualize the distribution of continuous variable across target variable

*# define continuous variable & plot*

continous\_features = ['age', 'chol', 'thalach', 'oldpeak','trestbps']

sns.pairplot(df[continous\_features + ['target']], hue='target')

<seaborn.axisgrid.PairGrid at 0x220aa6b0430>

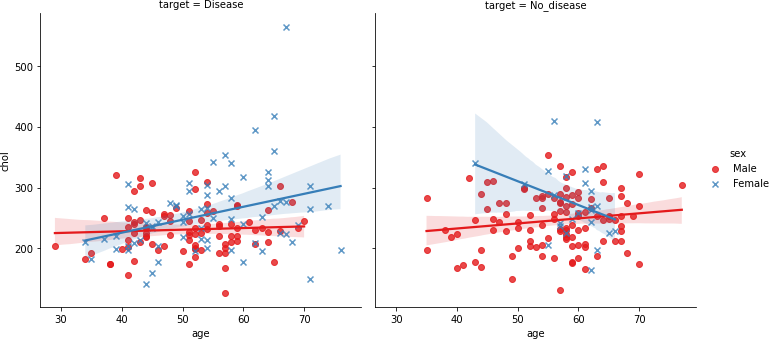


*# to understand the relationship between age and chol in each of the target based on sex.*

sns.lmplot(x="age", y="chol", hue="sex", col="target", markers=["o", "x"],

palette="Set1", data=df)

plt.show()



*# to understand the relationship between age and chol in each of the sex, based on target.*

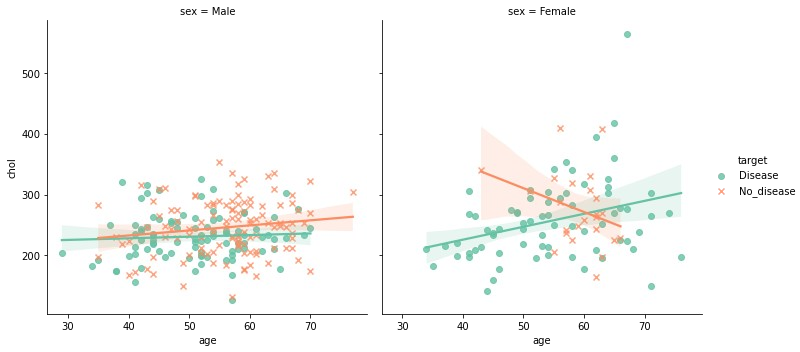
sns.lmplot(x="age",

y="chol", hue="target", col="sex",

*# row="target", # order=2,*

markers=["o", "x"], palette="Set2", data=df)

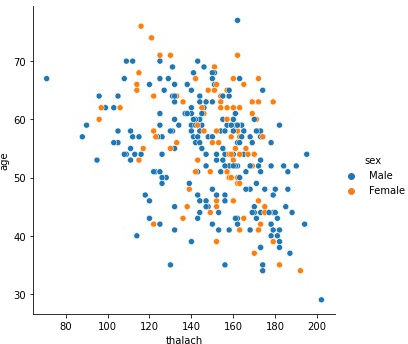
plt.show()



*# relation plot relplot*

sns.relplot(x='thalach', y = 'age', hue='sex', data=df )

<seaborn.axisgrid.FacetGrid at 0x220abb6ab80>



### Let's see the correlations

*# Correlation with Heatmap Visualization*

sns.set(style="white")

mask = np.zeros\_like(df.corr(), dtype=np.bool) mask[np.triu\_indices\_from(mask)] = True

fig, ax = plt.subplots(figsize=(5,5))

cmap = sns.diverging\_palette(255, 10, as\_cmap=True) sns.heatmap(df.corr(), mask=mask, annot=True, square=True, cmap=cmap,vmin=-1, vmax=1, ax=ax)

bottom, top = ax.get\_ylim() ax.set\_ylim(bottom+0.5, top-0.5)

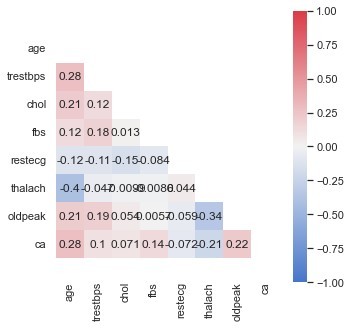
C:\Users\Lenovo\AppData\Local\Temp\ipykernel\_8784\2831669817.py:3: DeprecationWarning: `np.bool` is a deprecated alias for the builtin

`bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool\_` here.

Deprecated in NumPy 1.20; for more details and guidance:

https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations mask = np.zeros\_like(df.corr(), dtype=np.bool)

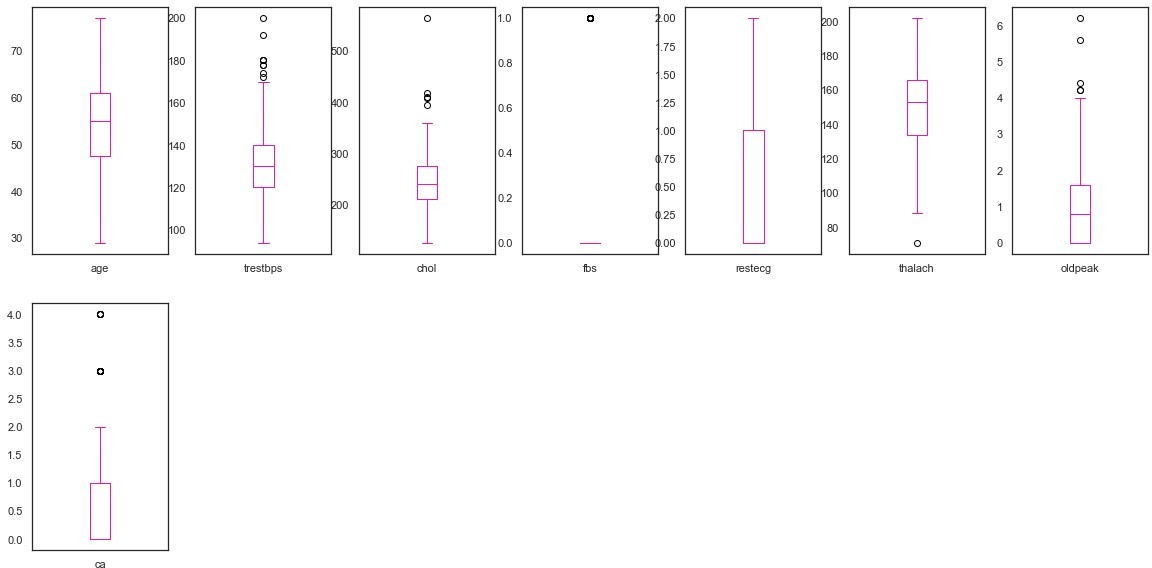
(8.5, -0.5)



# Check for outliers

*# to check outliers*

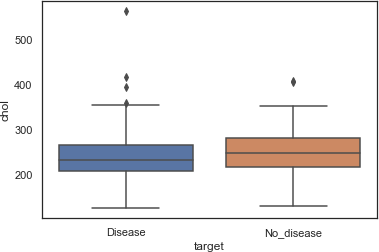
df.plot(kind='box', subplots=True, layout=(2,7), sharex=False,sharey=False, figsize=(20, 10), color='deeppink');



### Check outliers in each categorical features

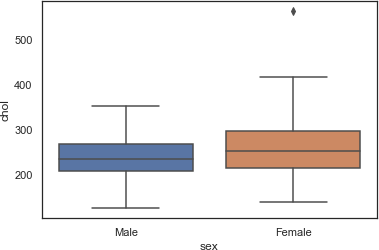
sns.boxplot(x='target', y='chol', data=df)

<AxesSubplot:xlabel='target', ylabel='chol'>



sns.boxplot(x='sex', y='chol', data=df)

<AxesSubplot:xlabel='sex', ylabel='chol'>

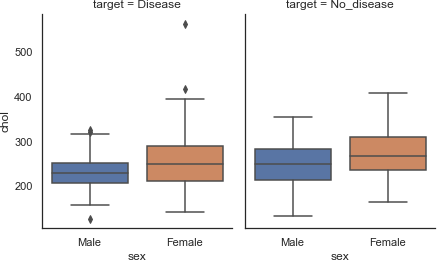


plt.figure(figsize=(15,10))

sns.catplot(x='sex', y='chol', col='target', data=df, kind='box', height=4, aspect=.8)

<seaborn.axisgrid.FacetGrid at 0x220abb8f700>

<Figure size 1080x720 with 0 Axes>



plt.figure(figsize=(15,10))

sns.catplot(x='sex', y='age', col='target', data=df, kind='box', height=4, aspect=.8)

<seaborn.axisgrid.FacetGrid at 0x220abd295b0>

<Figure size 1080x720 with 0 Axes>

