Nanditha.V
160119733133

Data Science and Big Data Analytics

Experiment 2: Exploratory Data Analysis

AIM: To do exploratory data analysis on heart disease UCI dataset.

DESCRIPTION:

Exploratory Data Analysis (EDA) is a pre-processing step to understand the data. There are numerous methods and steps in performing EDA, however, most of them are specific, focusing on either visualization or distribution, and are incomplete. Therefore, we have to understand, explore, and extract the information from the data to answer the questions or assumptions.

Data Set Explanations:

Here we will be using the dataset consisting of 303 patients with 14 features set.

Features explanation:

1. age (Age in years)
2. sex : (1 = male, 0 = female)
3. cp (Chest Pain Type): [0: asymptomatic, 1: atypical angina, 2: non-anginal pain, 3: typical angina]
4. trestbps (Resting Blood Pressure in mm/hg)
5. chol (Serum Cholesterol in mg/dl)
6. fps (Fasting Blood Sugar > 120 mg/dl): [0 = no, 1 = yes]
7. restecg (Resting ECG): [0: showing probable or definite left ventricular hypertrophy by Estes' criteria, 1: normal, 2: having ST-T wave abnormality]
8. thalach (maximum heart rate achieved)
9. exang (Exercise Induced Angina): [1 = yes, 0 = no]
10. oldpeak (ST depression induced by exercise relative to rest)
11. slope (the slope of the peak exercise ST segment): [0: downsloping; 1: flat; 2: upsloping]
12. ca [number of major vessels (0–3)
13. thal : [1 = normal, 2 = fixed defect, 3 = reversible defect]
14. target: [0 = disease, 1 = no disease]

CODE AND ANALYSIS:

1. Import and get to know the data

Nanditha.V
160119733133

In [2]:
```
# 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
```

In [3]:
```
df = pd.read_csv('/content/drive/My Drive/csv/github/heart.csv')
df.head(3)
```

Out[3]:

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

In [4]:
```
df.shape
```

Out[4]:
```
(303, 14)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   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

Here we have 303 rows with 14 variables.

2.Data Cleaning

a) Check the data type**.**

The variables types are

- Binary: sex, fbs, exang, target
- Categorical: cp, restecg, slope, ca, thal
- Continuous: age, trestbps, chol, thalac, oldpeak

Nanditha.V
160119733133

In [7]:
```
# to know the type of variable
df.nunique()
```

Out[7]:
```
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
```

In [8]:
```
df.dtypes
```

Out[8]:
```
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
```

Note here that the binary and categorical variable are classified as different integer type by python. We will need to change them to ‘object’ type.

In [9]:
```
# change the categorical type to categorical variables
df['sex'] = df['sex'].astype('object')
df['cp'] = df['cp'].astype('object')
df['fbs'] = df['fbs'].astype('object')
df['restecg'] = df['restecg'].astype('object')
df['exang'] = df['exang'].astype('object')
df['slope'] = df['slope'].astype('object')
df['ca'] = df['ca'].astype('object')
df['thal'] = df['thal'].astype('object')
df.dtypes
```

Out[9]:
```
age            int64
sex           object
cp            object
trestbps       int64
chol           int64
fbs           object
restecg       object
thalach        int64
exang         object
oldpeak      float64
slope         object
ca            object
thal          object
target         int64
dtype: object
```

b. Check for the data characters mistakes

Nanditha.V
160119733133

1.Feature 'ca' ranges from 0–3, however, df.nunique**()** listed 0–4. So lets find the '4' and change them to NaN.

```
In [10]: df['ca'].unique()

Out[10]: array([0, 2, 1, 3, 4], dtype=object)
```

```
In [11]: # to count the number in of each category decending order
         df.ca.value_counts()

Out[11]: 0    175
         1     65
         2     38
         3     20
         4      5
         Name: ca, dtype: int64
```

```
In [12]: # to check missing values
         df.isnull().sum()

Out[12]: 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
```

c) Check for missing values and replace them


```
In [12]: # to check missing values
         df.isnull().sum()

Out[12]: 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
```


d) Statistics summary

Nanditha.V
160119733133

In [13]:
```
# 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"})
```

In [14]:
```
# to know the basic stats
df.describe()
```

Out[14]:

	age	trestbps	chol	thalach	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	131.623762	246.264026	149.646865	1.039604
std	9.082101	17.538143	51.830751	22.905161	1.161075
min	29.000000	94.000000	126.000000	71.000000	0.000000
25%	47.500000	120.000000	211.000000	133.500000	0.000000
50%	55.000000	130.000000	240.000000	153.000000	0.800000
75%	61.000000	140.000000	274.500000	166.000000	1.600000
max	77.000000	200.000000	564.000000	202.000000	6.200000

BARPLOTS:
target variable distribution

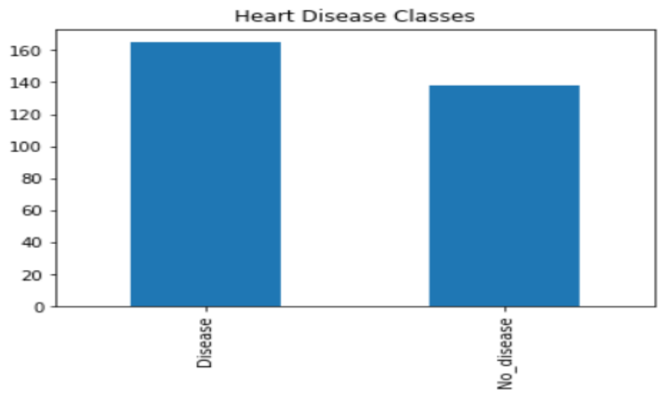
In [17]:
```
df.columns
```

Out[17]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [35]:
```
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
```

Out[35]:
```
Text(0.5, 1.0, 'Heart Disease Classes')
```



There are more diseased than healthy patients.

Age variable distribution

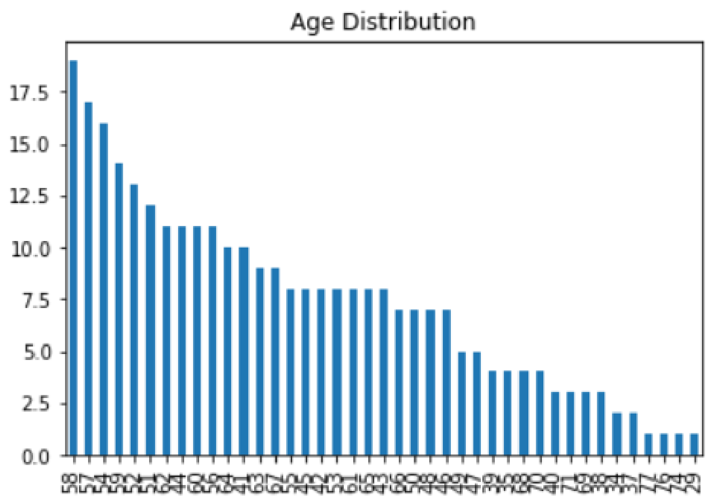
Nanditha.V
160119733133

In [23]:

```
# print(df.age.value_counts())
df['age'].value_counts().plot(kind='bar').set_title('Age Distribution')
```

Out[23]:

```
Text(0.5, 1.0, 'Age Distribution')
```



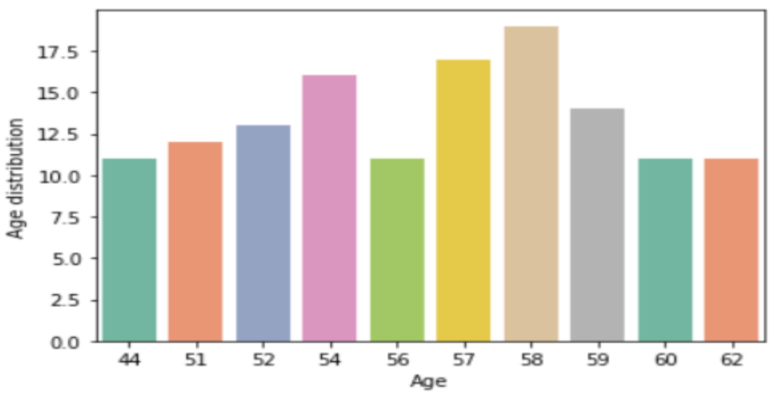
In [24]:

```
# 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
44    11
60    11
56    11
Name: age, dtype: int64
```

Out[24]:

```
Text(0, 0.5, 'Age distribution')
```



The age are normally distributed. Most of the patients are in the age between 50s to 60s. Let's take a quick look basic stats. The mean age is about 54 years with ±9.08 std, the youngest is at 29 and the oldest is at 77.

Nanditha.V
160119733133

In [ ]:

```
# to know the youngest or oldest in age
print(min(df.age))
print(max(df.age))
print(df.age.mean())
```

```
29
77
54.366336633663366
```

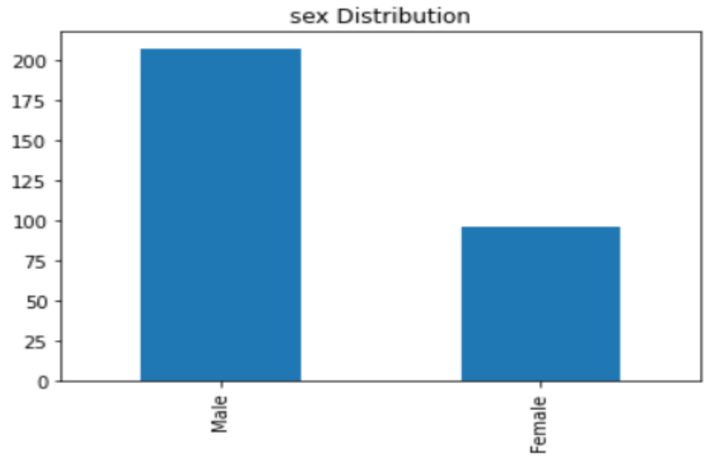
In [25]:

```
print(df.sex.value_counts())
df['sex'].value_counts().plot(kind='bar').set_title('sex Distribution')
```

```
Male      207
Female     96
Name: sex, dtype: int64
```

Out[25]:

```
Text(0.5, 1.0, 'sex Distribution')
```

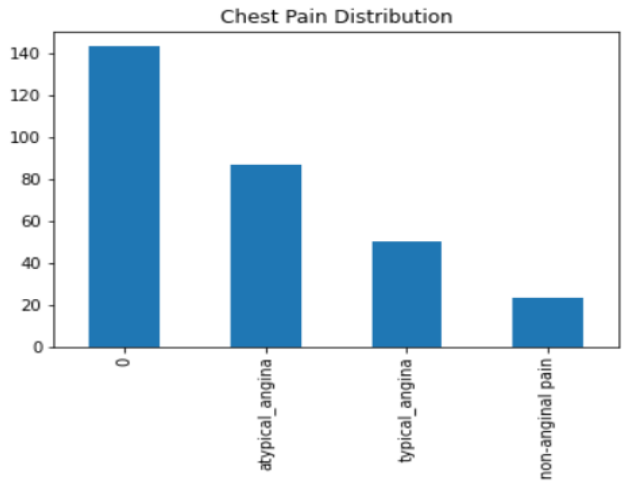


Nanditha.V
160119733133

In [26]:
```
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
```

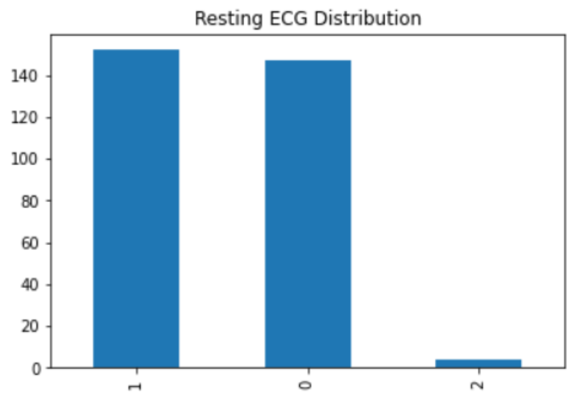
Out[26]:
```
Text(0.5, 1.0, 'Chest Pain Distribution')
```



In [43]:
```
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
```

Out[43]:
```
Text(0.5, 1.0, 'Resting ECG Distribution')
```

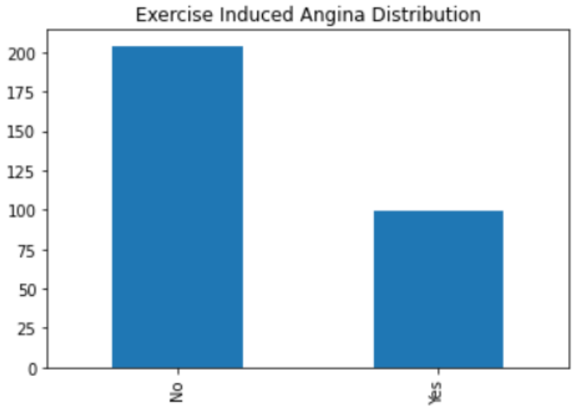


Nanditha.V
160119733133

In [32]:
```
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
```

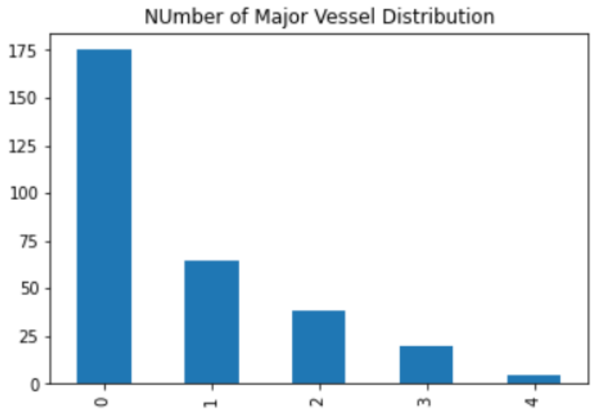
Out[32]:
```
Text(0.5, 1.0, 'Exercise Induced Angina Distribution')
```



In [33]:
```
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
```

Out[33]:
```
Text(0.5, 1.0, 'NUmber of Major Vessel Distribution')
```

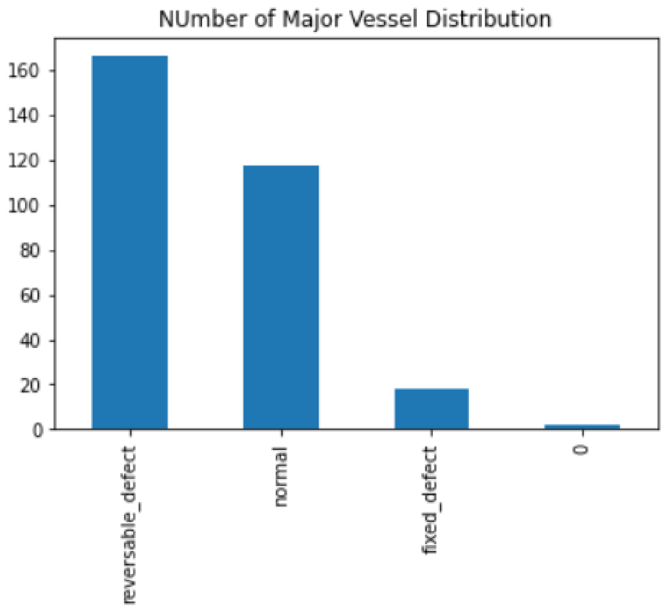


Nanditha.V
160119733133

In [34]:
```
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
```

Out[34]:
```
Text(0.5, 1.0, 'NUmber of Major Vessel Distribution')
```

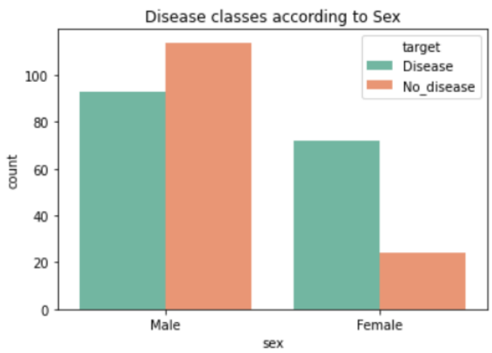


Visualize categorical data distribution -COUNTPLOTS

Gender distribution according to target variable

In [38]:
```
sns.countplot(x='sex', hue='target', data=df, palette='Set2').set_title('Disease classes according to Sex')
```

Out[38]:
```
Text(0.5, 1.0, 'Disease classes according to Sex')
```



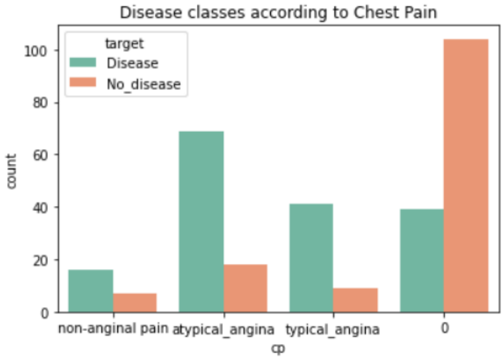
From the bar graph, we can observe that among disease patients, male are higher than female.

Chest pain distribution according to target variable

Nanditha.V
160119733133

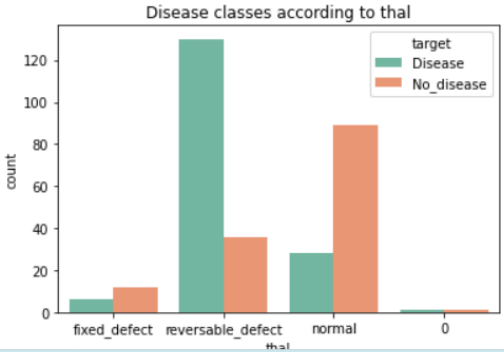
In [40]:
```
sns.countplot(x='cp', hue='target', data=df, palette='Set2').set_title('Disease classes according to Chest Pain')
```

Out[40]: Text(0.5, 1.0, 'Disease classes according to Chest Pain')



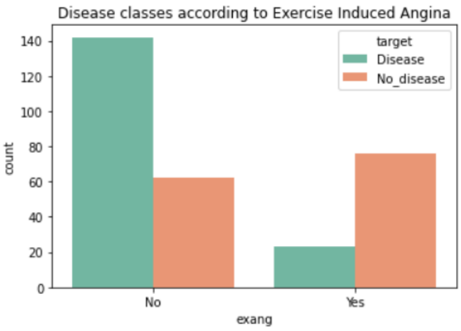
In [41]:
```
sns.countplot(x='thal', hue='target', data=df, palette='Set2').set_title('Disease classes according to thal')
```

Out[41]: Text(0.5, 1.0, 'Disease classes according to thal')



In [44]:
```
sns.countplot(x='exang', hue='target', data=df, palette='Set2').set_title('Disease classes according to Exercise Induced Angina')
```

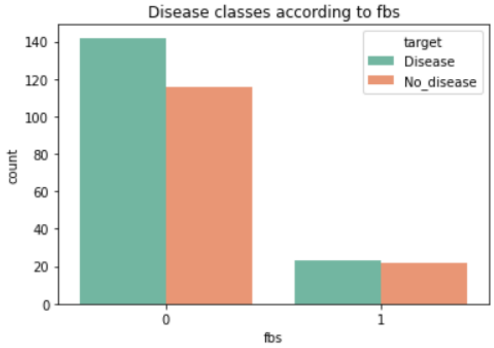
Out[44]: Text(0.5, 1.0, 'Disease classes according to Exercise Induced Angina')



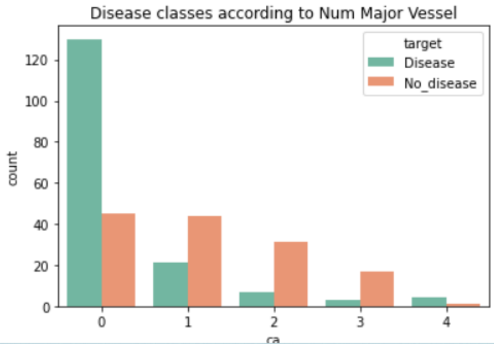
Fasting blood sugar distribution according to target variable

Nanditha.V
160119733133

In [45]:
```
sns.countplot(x='fbs', hue='target', data=df, palette='Set2').set_title('Disease classes according to fbs')
```
Out[45]:
```
Text(0.5, 1.0, 'Disease classes according to fbs')
```



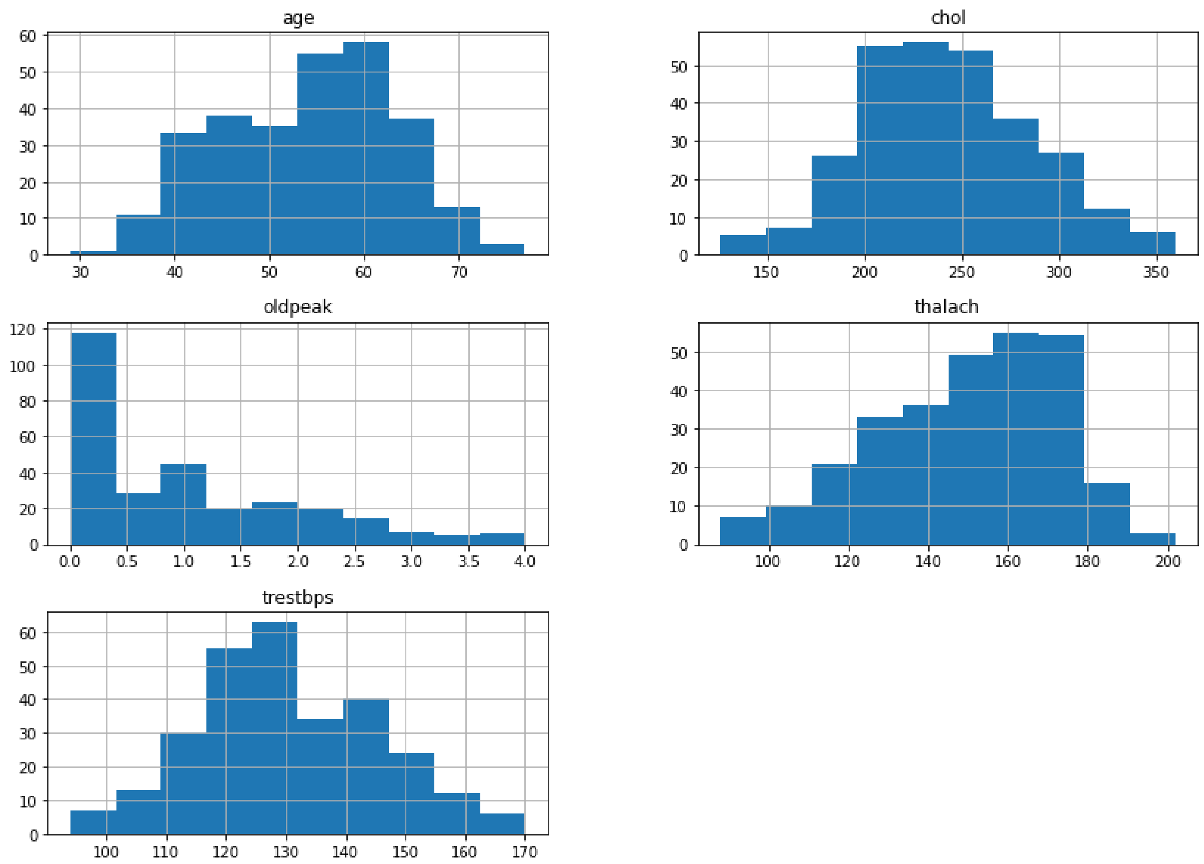
In [46]:
```
sns.countplot(x='ca', hue='target', data=df, palette='Set2').set_title('Disease classes according to Num Major Vessel')
```
Out[46]:
```
Text(0.5, 1.0, 'Disease classes according to Num Major Vessel')
```



Looking at the bar graph above, its raised a question of higher number of healthy subject having typical_angina. Or in other word, most of the healthy subject having chest pain.

Fasting blood sugar or fbs is a diabetes indicator with fbs >120 mg/d is considered diabetic (True class). Here, we observe that the number for class true, is lower compared to class false. However, if we look closely, there are higher number of heart disease patient without diabetes. This provide an indication that fbs might not be a strong feature differentiating between heart disease an non-disease patient.

Distribution plot on continuous variables.



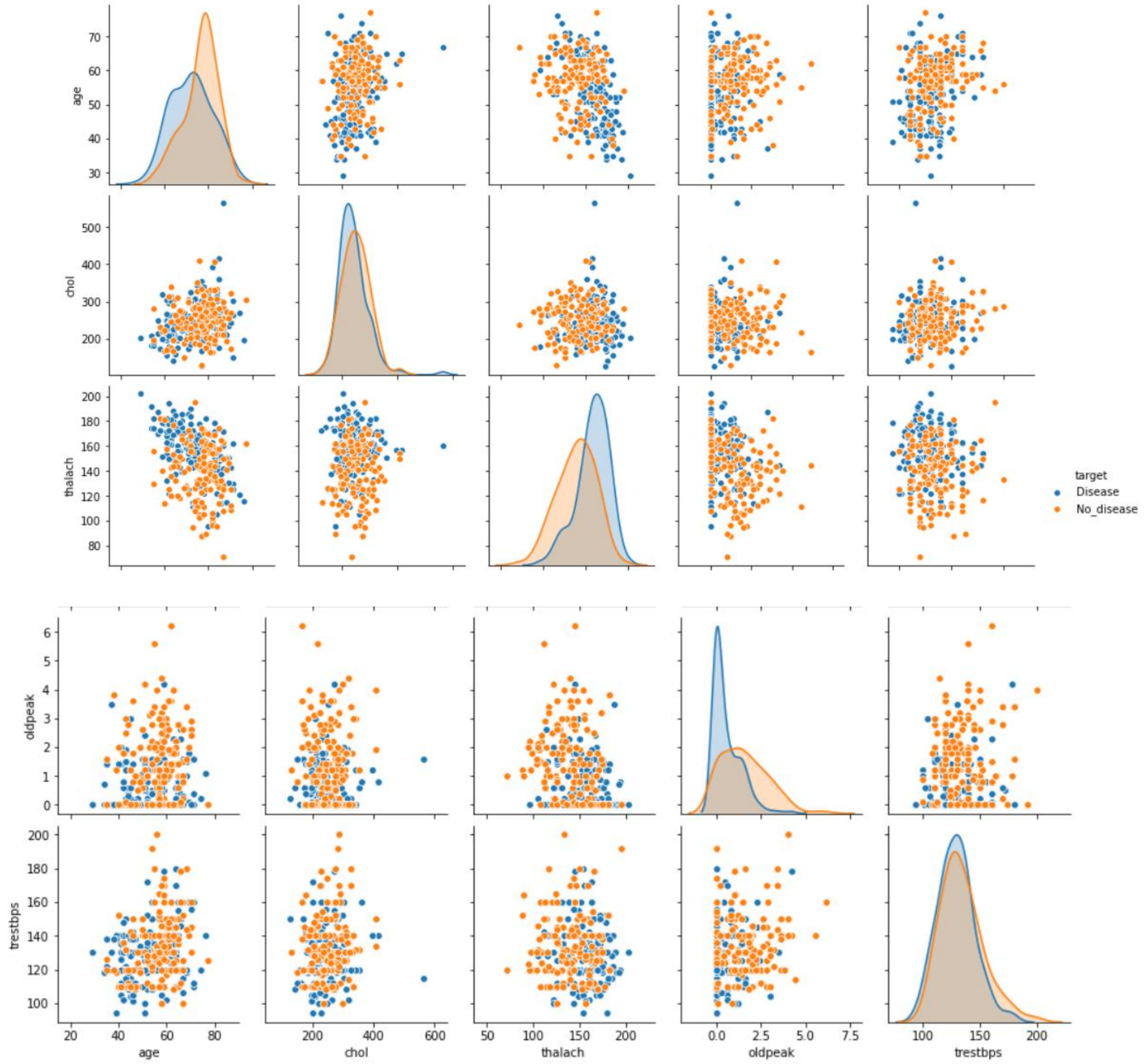
- normal distribution for: age, trestbps and almost for chol
- oldpeak is left-skewed
- thalac is right-skewed

Visualize the distribution of continuous variable across target variable-PAIRPLOTs

Nanditha.V
160119733133

In [55]:
```
# define continuous variable & plot
continous_features = ['age', 'chol', 'thalach', 'oldpeak','trestbps']
sns.pairplot(df[continous_features + ['target']], hue='target')
```

Out[55]: <seaborn.axisgrid.PairGrid at 0x7f562cdf38d0>

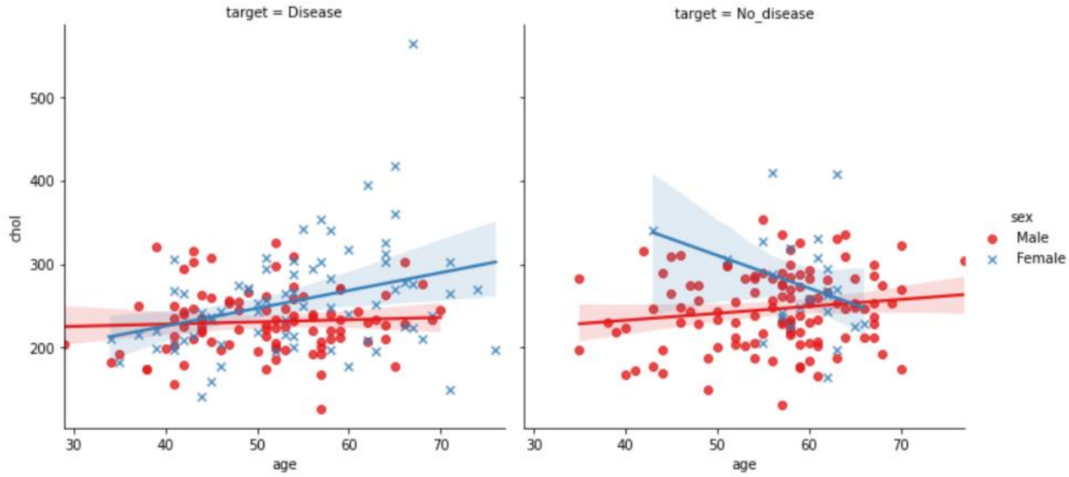


- oldpeak having a linear separation relation between disease and non-disease.
- thalach having a mild separation relation between disease and non-disease.
- Other features don't form any clear separation

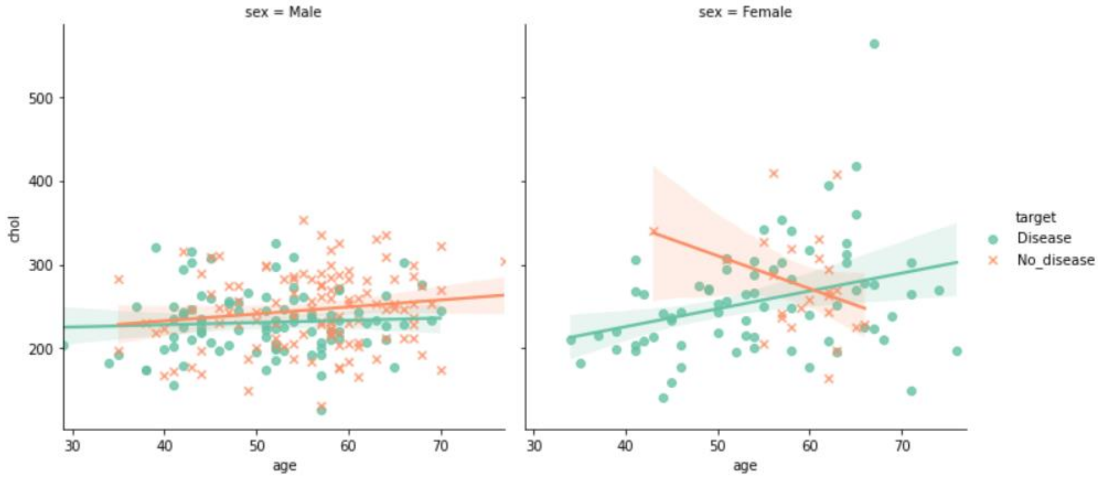
Lineplots:

Nanditha.V
160119733133

In [56]:
```
# 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()
```



In [ ]:
```
# 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()
```



correlation

Nanditha.V
160119733133

In [ ]:

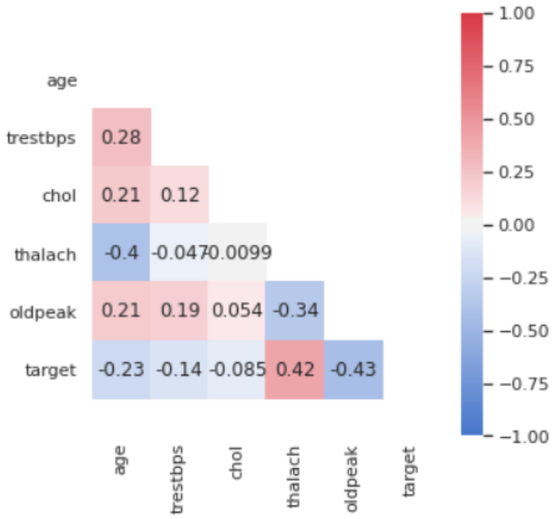
```
# Correlation with Heatmap Visualization
sns.set(style="white")
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

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)
```

Out[ ]:

```
(6.5, -0.5)
```



- ‘cp’, ‘thalach’, ‘slope’ shows good positive correlation with target
- ‘oldpeak’, ‘exang’, ‘ca’, ‘thal’, ‘sex’, ‘age’ shows a good negative correlation with target
- ‘fbs’ ‘chol’, ‘trestbps’, ‘restecg’ has low correlation with our target