Capstone

July 16, 2024

[ ]:
```
#Import libraries such as Pandas, matplotlib, NumPy, and seaborn and load the dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/usr/local/lib/python3.10/site-packages/scipy/__init__.py:155: UserWarning: A
NumPy version >=1.18.5 and <1.26.0 is required for this version of SciPy
(detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[ ]:
```
df = pd.read_excel('data.xlsx')
```

[ ]:
```
1.Preliminary analysis:
1.Perform preliminary data inspection and report the findings as the structure of the data, missing values, duplicates etc.
2.Based on the findings from the previous question remove duplicates (if any) ,treat missing values using appropriate strategy.
```

[ ]:
```
# Understanding Data
```

[12]:
```
df.head()
```

[12]:

	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
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
4   0     2       1
```

[8]:
```
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
```

[14]:
```
#Duplicate values
df[df.duplicated]
```

[14]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
164   38    1   2       138   175    0        1      173      0      0.0

      slope  ca  thal  target
164       2   4     2       1
```

[15]:
```
df.duplicated().sum()
```

[15]:
```
1
```

[ ]:
```
#Remove the duplicate
```

[6]:
```
df.drop_duplicates(inplace= True)
```

[ ]:
```
1.Get a preliminary statistical summary of the data. Explore the measures of␣
 ↪central tendencies and the spread of the data overall.
```

[9]:
```
df.describe()
```

[9]:

	age	sex	cp	trestbps	chol	fbs
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000

	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373
std	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000
75%	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

	thal	target
count	303.000000	303.000000
mean	2.313531	0.544554
std	0.612277	0.498835
min	0.000000	0.000000
25%	2.000000	0.000000
50%	2.000000	1.000000
75%	3.000000	1.000000
max	3.000000	1.000000

[ ]:
```
df.rename ({'cp' :'Chest_pain_type',
'trestbps':'resting_blood_pressure',
'chol' : 'cholestoral',
'fbs' : 'fasting_blood_sugar',
'restecg' :'resting_ecg',
'thalach':'max_heart_rate',
'exang': 'exercise_induced_angina',
'oldpeak' : 'ST_depression',
'slope' :'ST_slope',
'ca': 'major_vessels',
'thal':'thalessimia'}, axis = 1, inplace= True)
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['age', 'sex', 'Chest_pain_type', 'resting_blood_pressure',
       'cholestoral', 'fasting_blood_sugar', 'resting_ecg', 'max_heart_rate',
       'exercise_induced_angina', 'ST_depression', 'ST_slope', 'major_vessels',
```

```
       'thalessimia', 'target'],
      dtype='object')
```

[9]:
```
2.Identify the data variables which might be categorical in nature. Describe␣
↪and explore these variables using appropriate tools e.g. count plot
```

```
  File "/tmp/ipykernel_76/785172915.py", line 1
    Identify the data variables which might be categorical in nature. Describe␣
 ↪and explore these variables using appropriate tools e.g. count plot
             ^
SyntaxError: invalid syntax
```

[25]:
```
cat_cols=['sex','Chest_pain_type',␣
 ↪'fasting_blood_sugar','exercise_induced_angina', 'ST_depression',␣
 ↪'ST_slope', 'thalessimia','target']
```

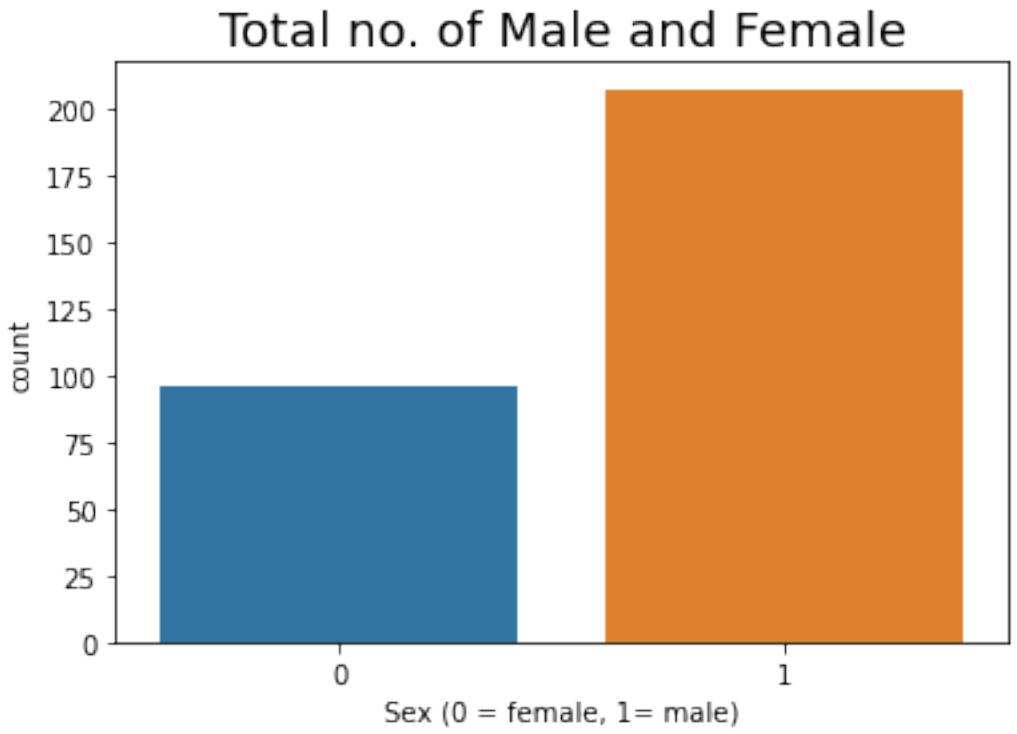
[12]:
```
df.thalessimia.value_counts()
```

[12]:
```
2    168
3    117
1     18
Name: thalessimia, dtype: int64
```

[ ]:
```
Note :
• thalessimia has 4 unique categories according to data however in description␣
 ↪there are only
3.
• there are 2 records which are identified as '0' ; these can be seen as␣
 ↪missing values and hence
need to be imputed.
• for imputation we can put in the category with modal value of '2'
```

[13]:
```
df.loc[df.thalessimia==0,'thalessimia']=2
```

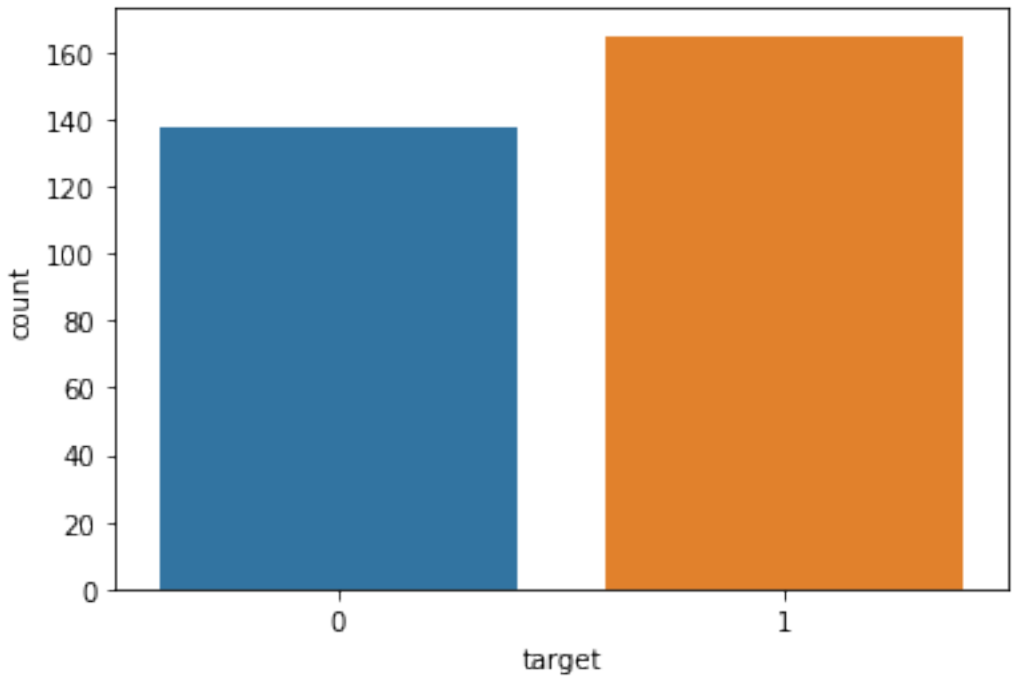
[17]:
```
sns.countplot(x = 'sex', data = df)
plt.title('Total no. of Male and Female', size=18)
plt.xlabel("Sex (0 = female, 1= male)")
plt.show()
```



[19]:
```
df.target.value_counts()
```

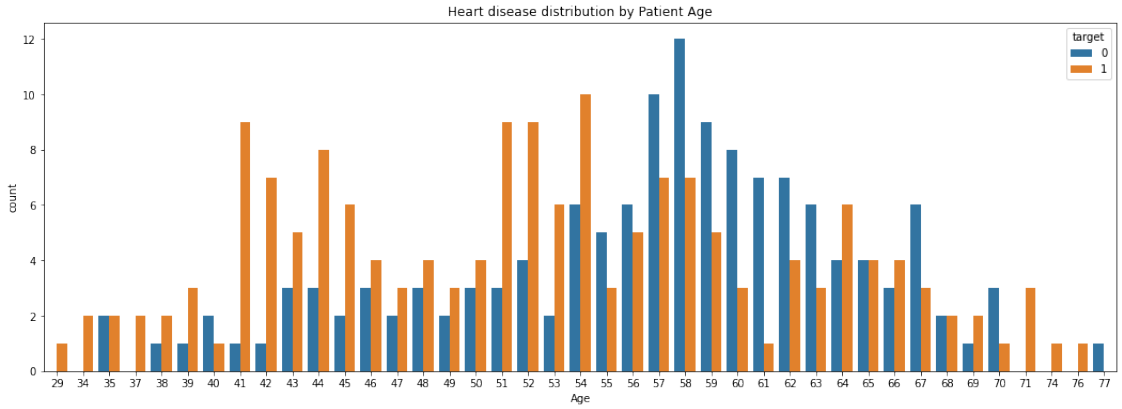
[19]:
```
1    165
0    138
Name: target, dtype: int64
```

[23]:
```
sns.countplot(x='target',data=df)
plt.show()
```



[ ]:
```
3.Study the occurence of CVD across age
```

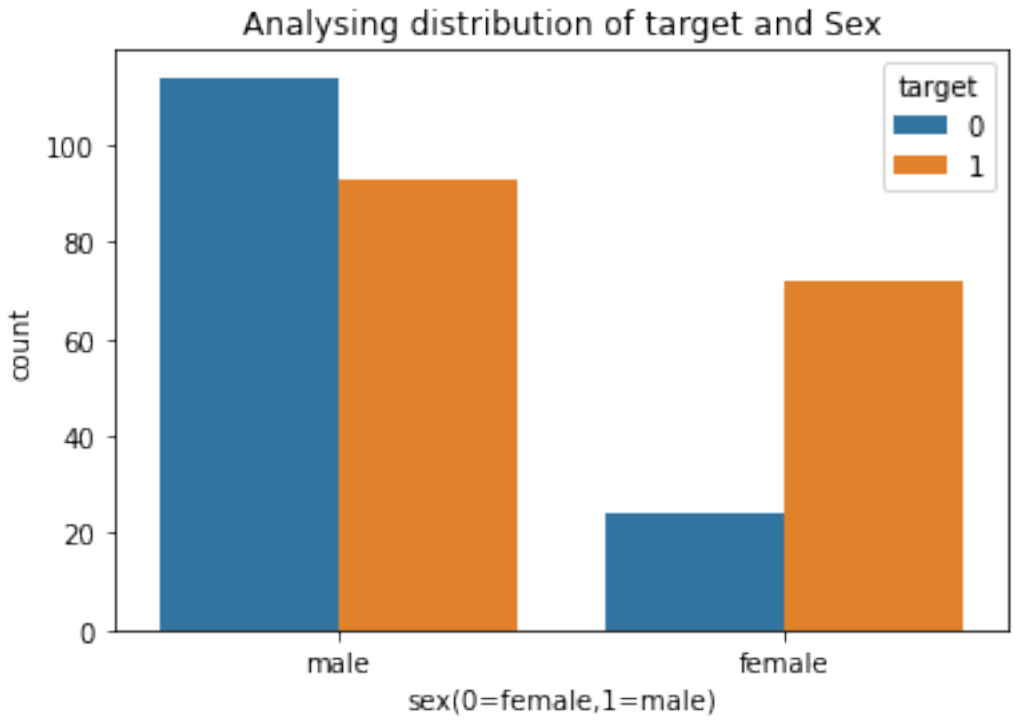
[41]:
```
plt.figure(figsize=(18,6))
sns.countplot(x=df['age'],hue = df['target'])
plt.title('Heart disease distribution by Patient Age')
plt.xlabel('Age')
plt.show()
```



[ ]:
```
4.Study the composition of overall patients w.r.t . Gender.
```

[51]:
```
df.loc[df.sex == 0, 'sex']='female'
df.loc[df.sex == 1, 'sex']='male'
```

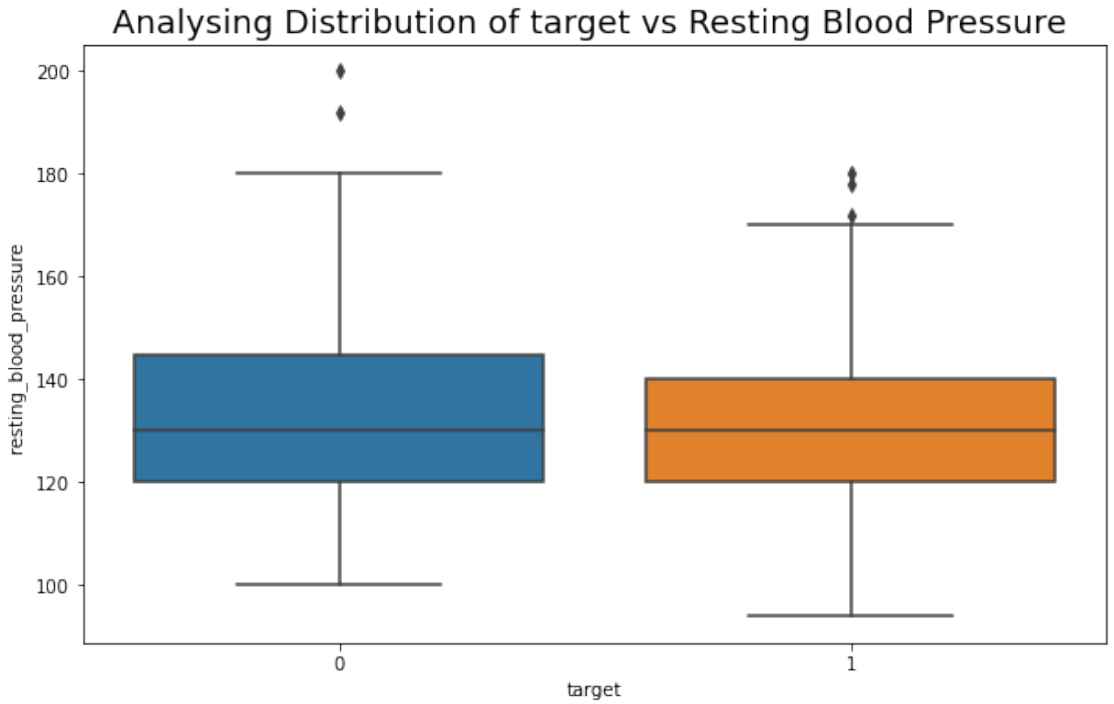
[89]:
```
plt.title('Analysing distribution of target and Sex')
sns.countplot(x=df.sex,hue=df['target'])
plt.xlabel('sex(0=female,1=male)')
plt.show()
```



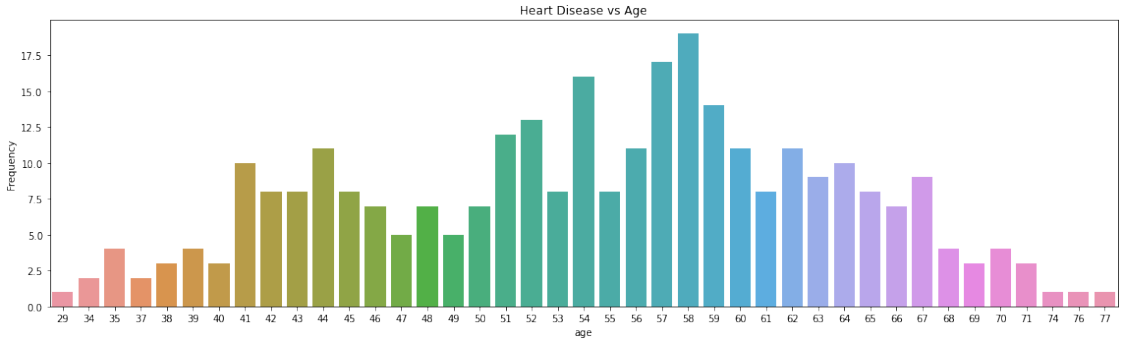
[53]:
```
5.Can we detect heart attack based on anomalies in Resting Blood Pressure of␣
↪the patient?
```

```
Object `patient` not found.
```

[90]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='target',y='resting_blood_pressure ',data=df)
plt.title('Analysing Distribution of target vs Resting Blood Pressure ',size=18)
plt.show()
```



[11]:
```
plt.figure(figsize=(16,5))
sns.countplot(x=df['age'])
plt.title('Heart Disease vs Age')
plt.tight_layout()
plt.xlabel('age')
plt.ylabel('Frequency')
plt.show()
```

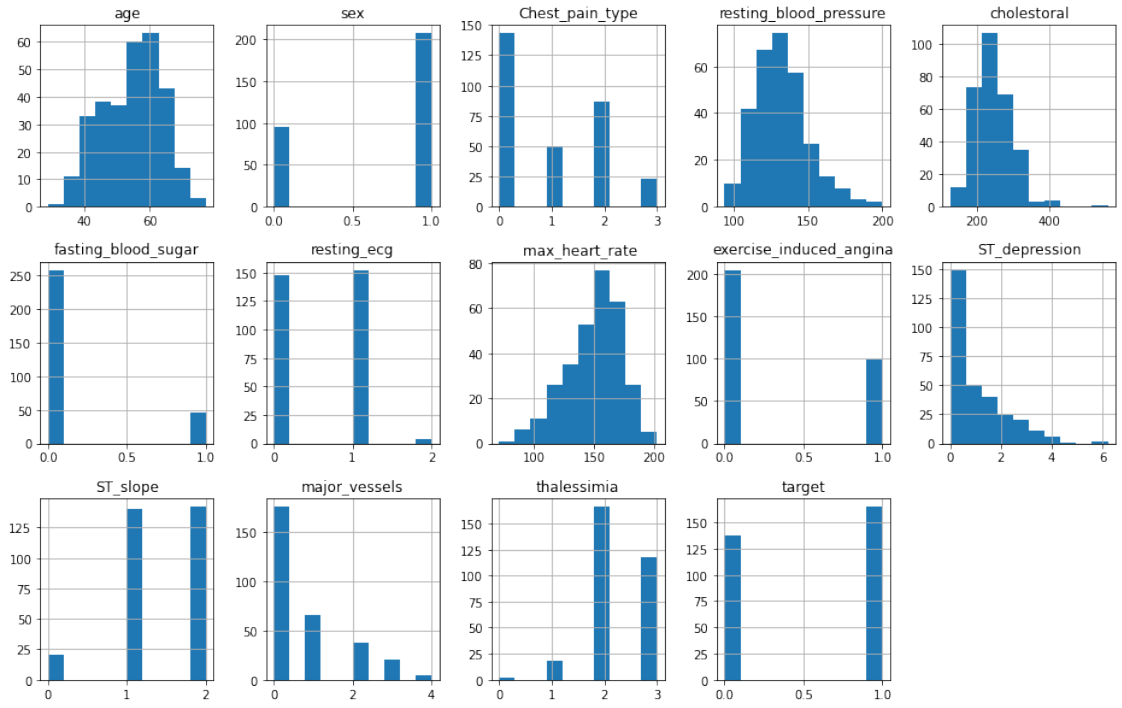


[ ]:
```
Note :
    the chances of heart attack acorss age at intermittent peaks
    The tendency of heart aatack increase after 40
```

```
    age group 51-54 and 57-59 have the highest chances of heart attack
```

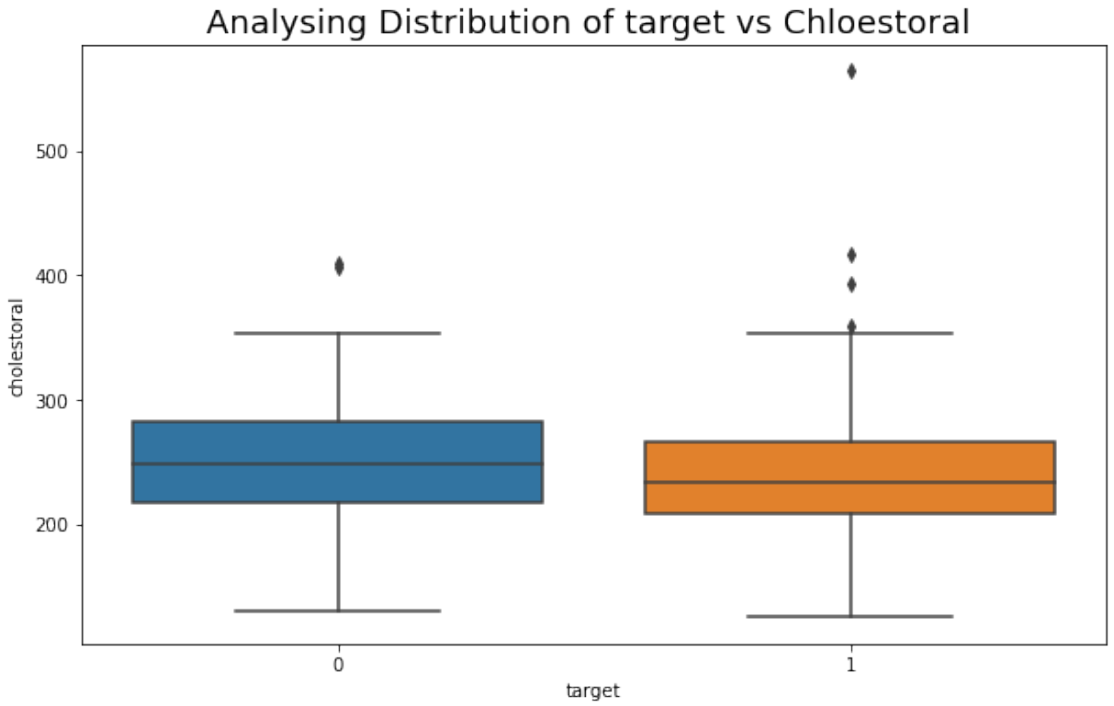
[21]:
```
df.hist(layout=(3,5),figsize=(16,10))
print('Data Distribution')
```

```
Data Distribution
```

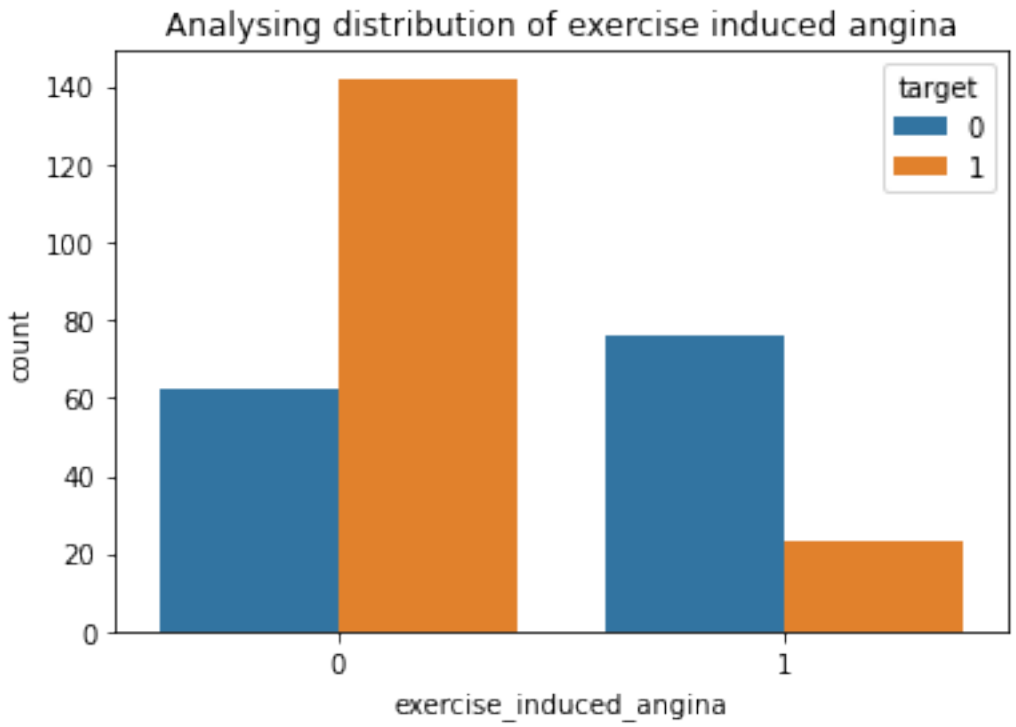


[ ]:
```
Describe the relationship between Cholesterol levels and our target variable.
```

[23]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='target',y='cholestoral',data=df)
plt.title('Analysing Distribution of target vs Chloestoral ',size=18)
plt.show()
```



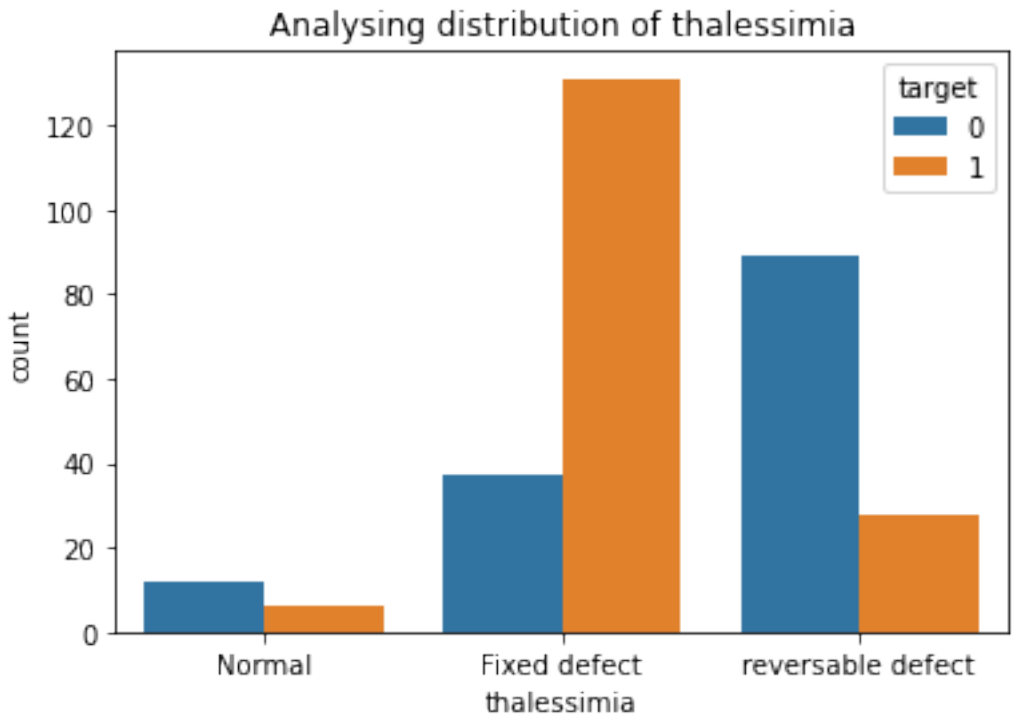
[9]:
```
plt.title('Analysing distribution of exercise induced angina')
sns.countplot(x=df.exercise_induced_angina,hue=df['target'])
plt.xlabel('exercise_induced_angina')
plt.show()
```



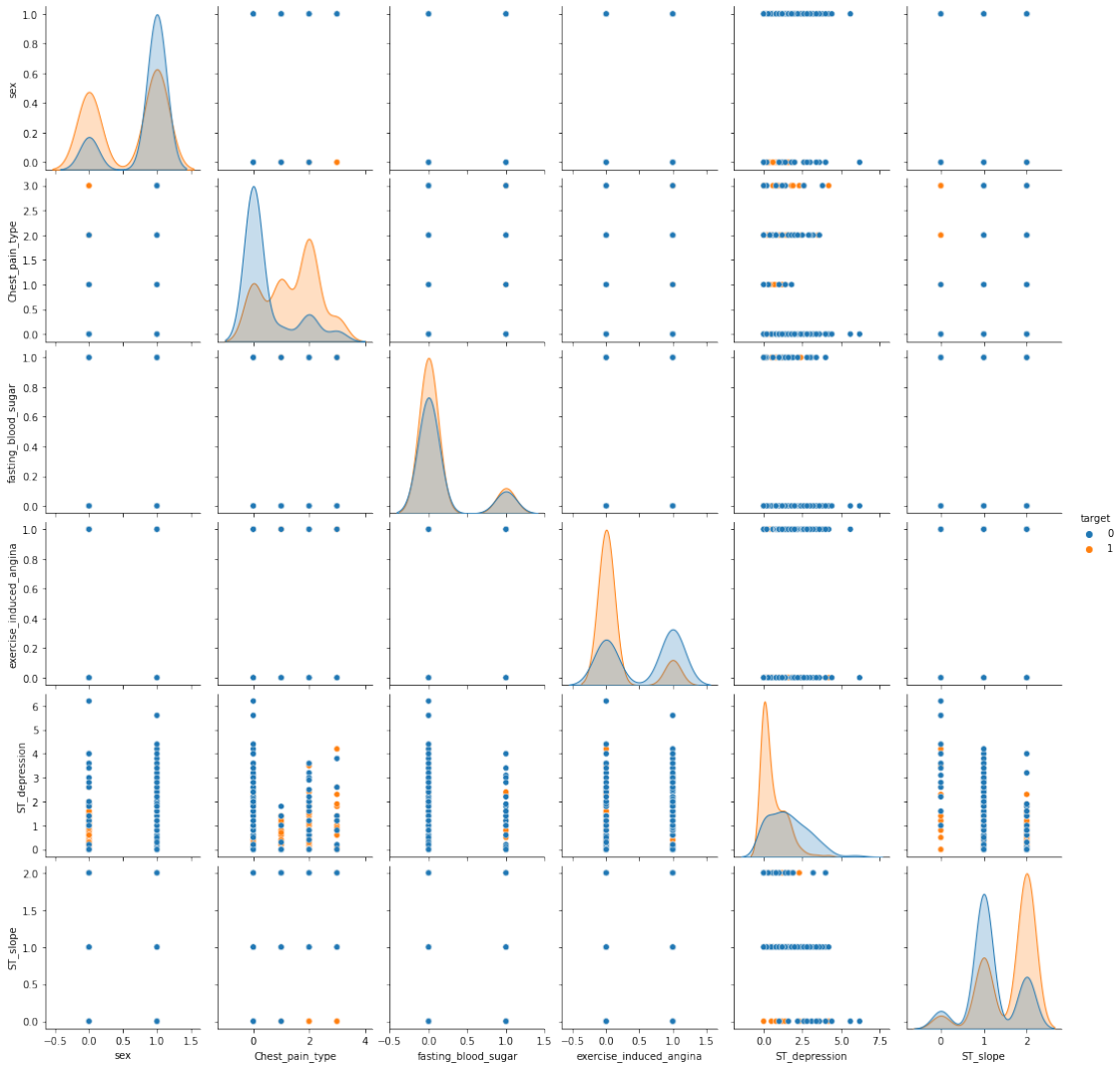
[ ]:
```
Is thalessimia a major cause of CVD ?
```

[15]:
```
df.loc[df.thalessimia == 1, 'thalessimia']='Normal'
df.loc[df.thalessimia == 2, 'thalessimia']='Fixed defect'
df.loc[df.thalessimia == 3, 'thalessimia']='reversable defect'
```

[16]:
```
plt.title('Analysing distribution of thalessimia')
sns.countplot(x=df.thalessimia,hue=df['target'])
plt.xlabel('thalessimia')
plt.show()
```



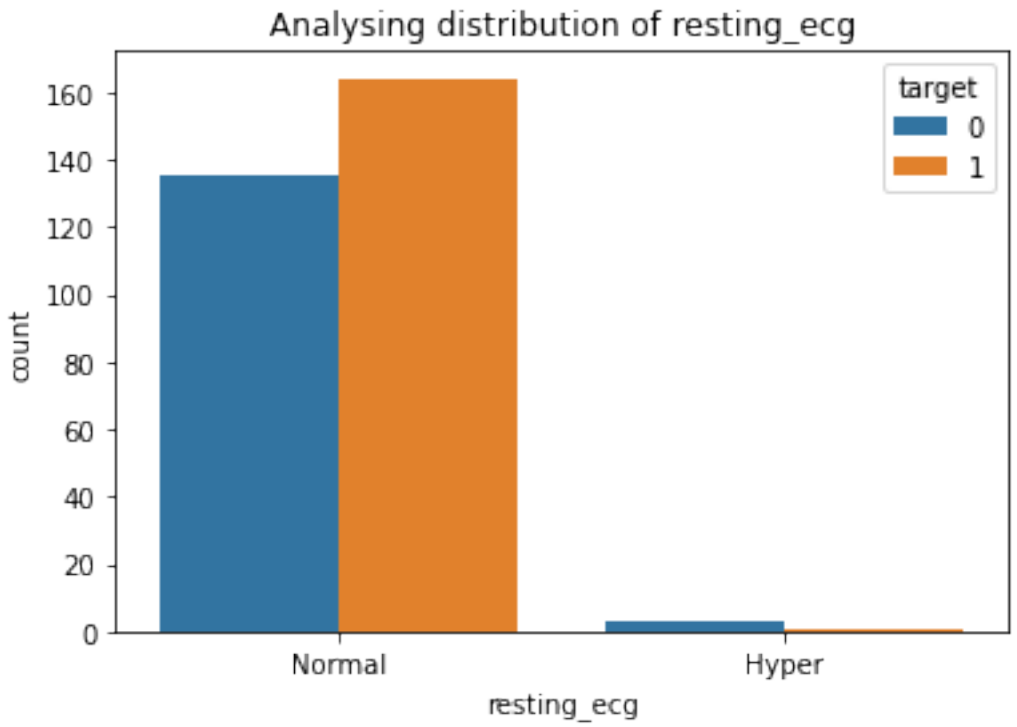
[30]:
```
sns.pairplot(df[cat_cols] , hue ='target')
plt.show()
```



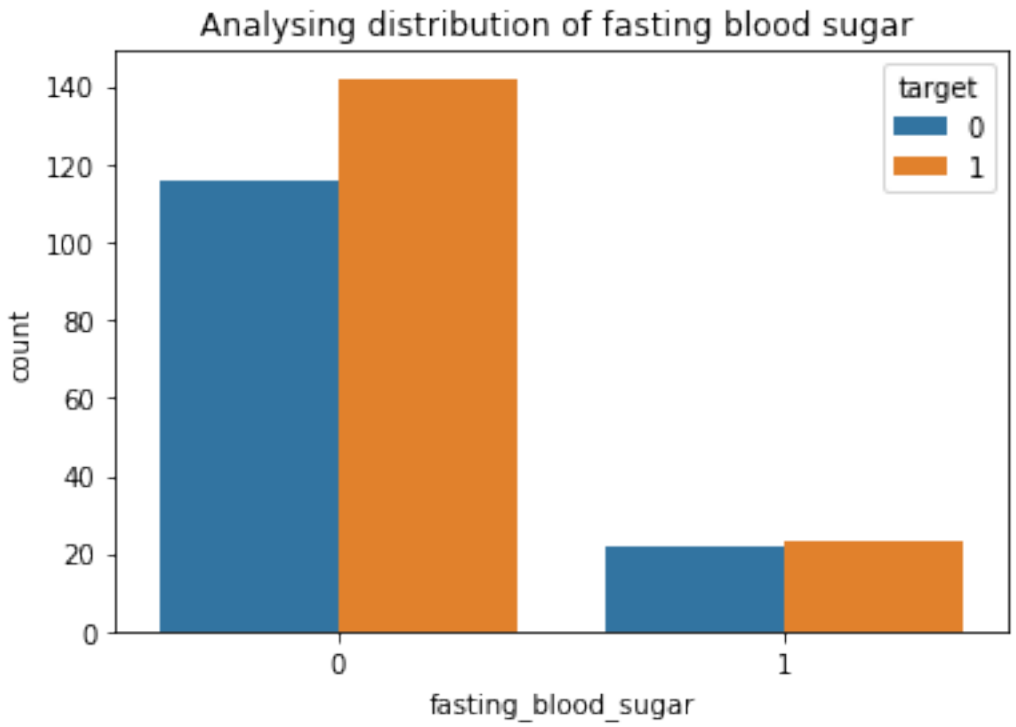
[ ]:
```
How are the other factors determining the occurrence of CVD?
```

```
Object `CVD` not found.
```

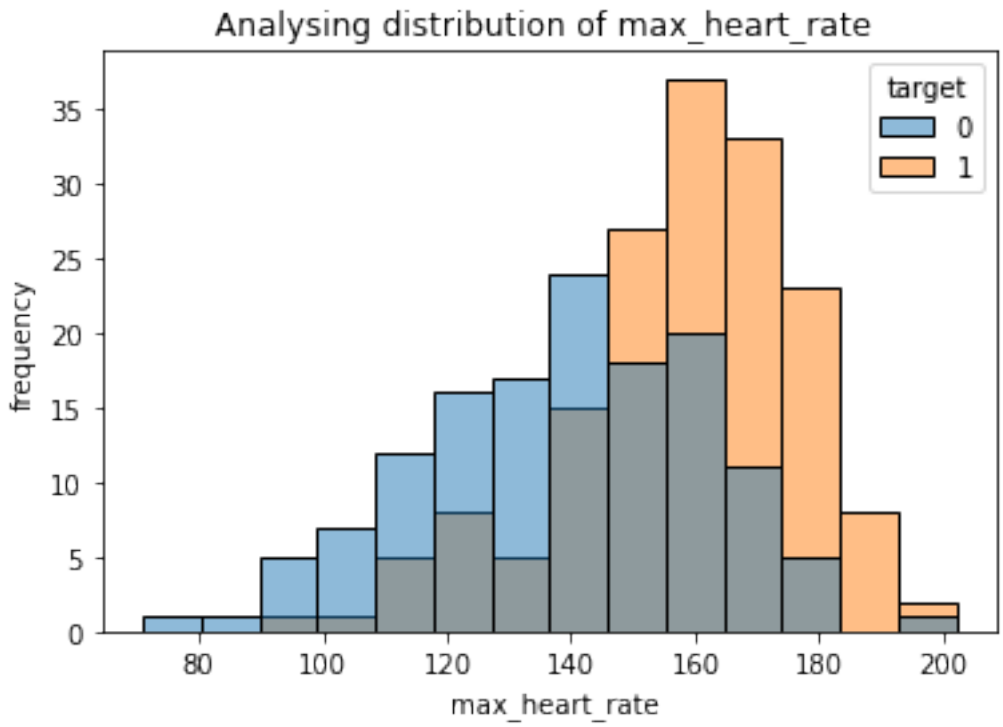
[51]:
```
plt.title('Analysing distribution of resting_ecg')
sns.countplot(x=df.resting_ecg,hue=df['target'])
plt.xlabel('resting_ecg')
plt.show()
```



[50]:
```
plt.title('Analysing distribution of fasting blood sugar ')
sns.countplot(x=df.fasting_blood_sugar,hue=df['target'])
plt.xlabel('fasting_blood_sugar')
plt.show()
```



[67]:
```
plt.title('Analysing distribution of max_heart_rate ')
sns.histplot(x=df[' max_heart_rate'],hue=df['target'])
plt.xlabel(' max_heart_rate')
plt.ylabel('frequency')
plt.show()
```



[ ]:
```
Build a baseline model to predict using a Logistic Regression and explore the␣
↪results.
```

[12]:
```
from sklearn.model_selection import train_test_split as split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
df_dummy = pd.get_dummies(df)
df_dummy.columns = df_dummy.columns.str.replace(' ','_')
train, test = split(df_dummy, test_size = .30, random_state = 12)
train.shape
train.head(2)
X_train = train.drop('target', axis = 1)
Y_train = train.target
X_test = test.drop('target', axis = 1)
Y_test = test.target
lr = LogisticRegression()
lr.fit(X_train,Y_train)
pred = lr.predict(X_test)
accuracy_score(y_true = Y_test,y_pred = pred)
print(classification_report(y_true=Y_test,y_pred = pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.80      0.82        45
```

```
           1       0.81      0.85      0.83        46

    accuracy                           0.82        91
   macro avg       0.82      0.82      0.82        91
weighted avg       0.82      0.82      0.82        91


/usr/local/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:460:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[13]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_test, pred))
```

```
[[36  9]
 [ 7 39]]
```

[ ]:
```

```