Cardio

February 9, 2023

0.0.1 Course-end Machine Leanring project based on cardiovascular disease.

[13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[14]:
```
df = pd.read_excel('cardio.xlsx')
df
```

[14]:

	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
..	…	…	..	…	…	…	…	…	…	…	…	..	…	…
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

[303 rows x 14 columns]

0.0.2 To be performed: Part 1

- perform data inspection
- remove duplicate data and treat missing values

[15]:
```
df.shape
```

[15]:
```
(303, 14)
```

[16]:
```
df.describe()
```

[16]:

	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

[17]:
```
df.isnull().any()
```

[17]:
```
age         False
sex         False
cp          False
trestbps    False
chol        False
```

```
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

[18]:
```
duplicate = df[df.duplicated()]
print(duplicate)
df.drop_duplicates(inplace = True)
```

```
     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
```

[20]:
```
df.shape
```

[20]:
```
(302, 14)
```

0.0.3 Part 2: Graphs and speard of the dataset at hand.

[23]:
```
df.describe()
```

[23]:

	age	sex	cp	trestbps	chol	fbs	\
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	
mean	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	
std	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	152.500000	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  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
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
```

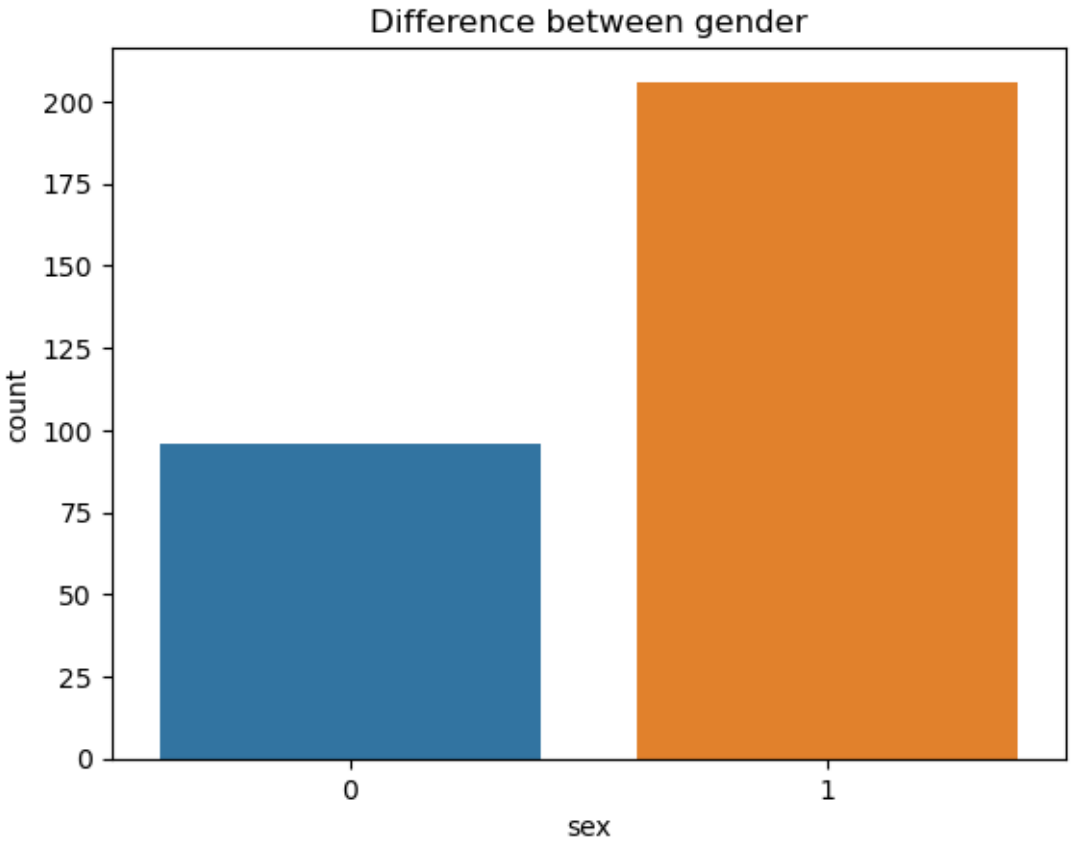
[26]:
```
## Categorcial data: Sex, cp, fbs, restecg, exang, slope, ca, thal, target

cat = df[['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal',␣
 ↪'target']]
```

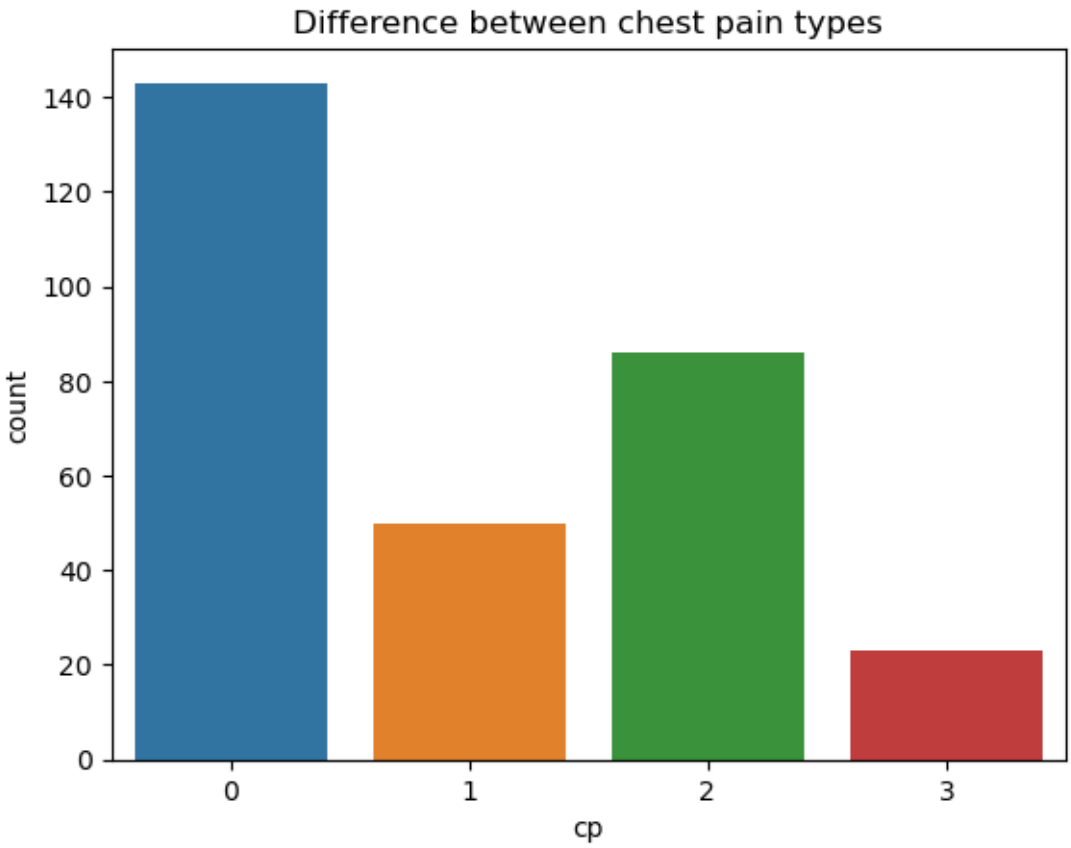
[27]:
```

```

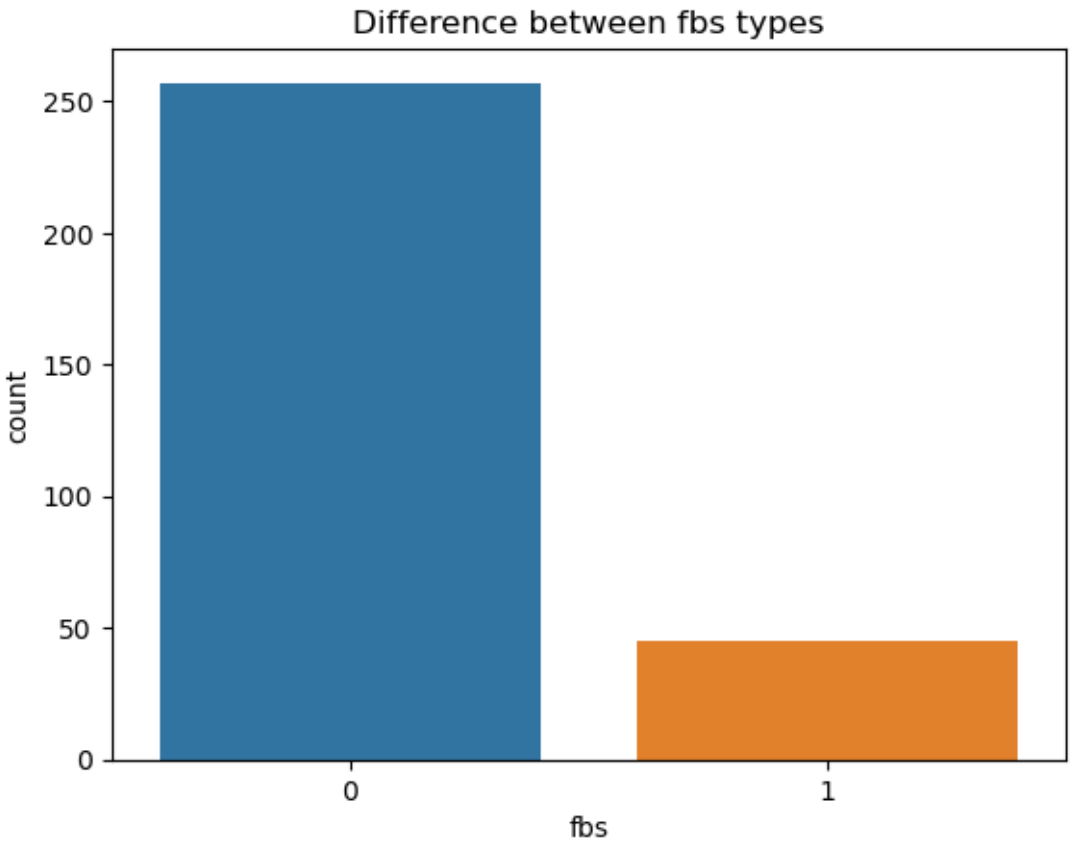
[27]:
```
     sex  cp  fbs  restecg  exang  slope  ca  thal  target
0      1   3    1        0      0      0   0     1       1
1      1   2    0        1      0      0   0     2       1
2      0   1    0        0      0      2   0     2       1
3      1   1    0        1      0      2   0     2       1
4      0   0    0        1      1      2   0     2       1
..   …  ..  …      …    …   …  ..  …     …
298    0   0    0        1      1      1   0     3       0
299    1   3    0        1      0      1   0     3       0
300    1   0    1        1      0      1   2     3       0
301    1   0    0        1      1      1   1     3       0
302    0   1    0        0      0      1   1     2       0

[302 rows x 9 columns]
```

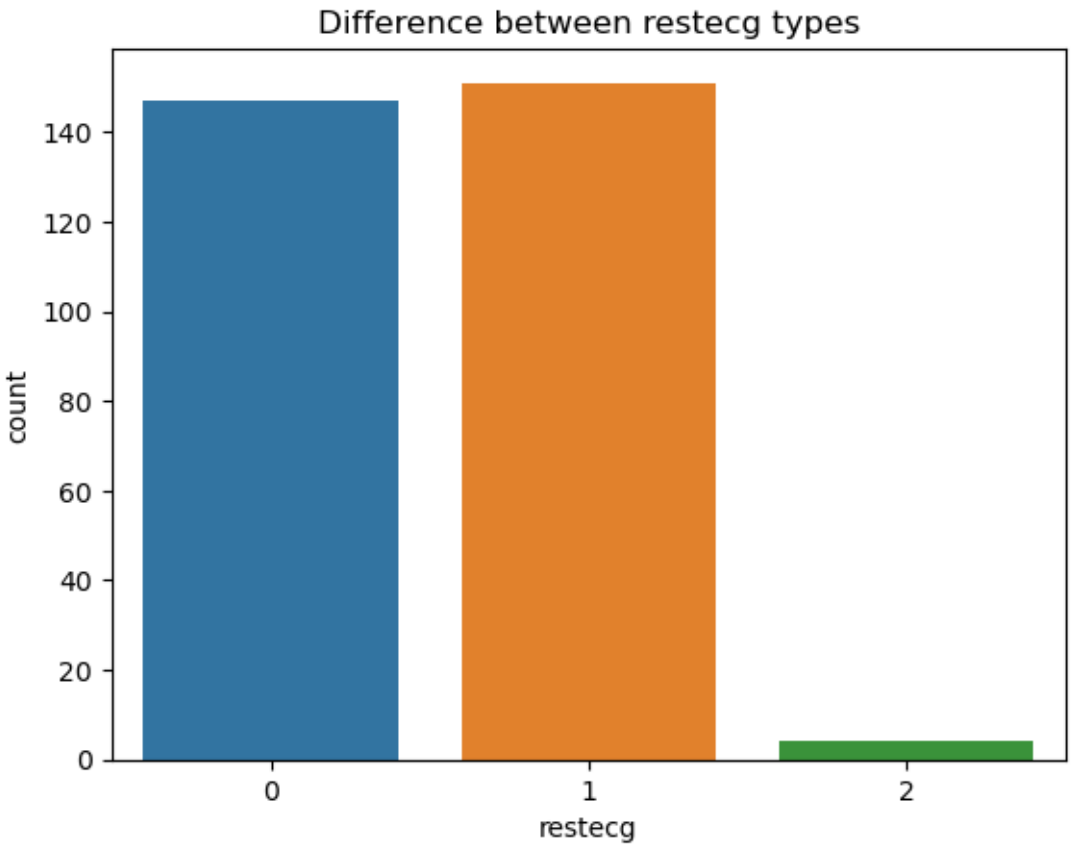
[342]:
```
import seaborn as sns
sns.countplot(data=df, x="sex")
plt.title('Difference between gender')
plt.show()

sns.countplot(data=df, x="cp")
plt.title('Difference between chest pain types')
plt.show()

sns.countplot(data=df, x="fbs")
plt.title('Difference between fbs types')
plt.show()
```

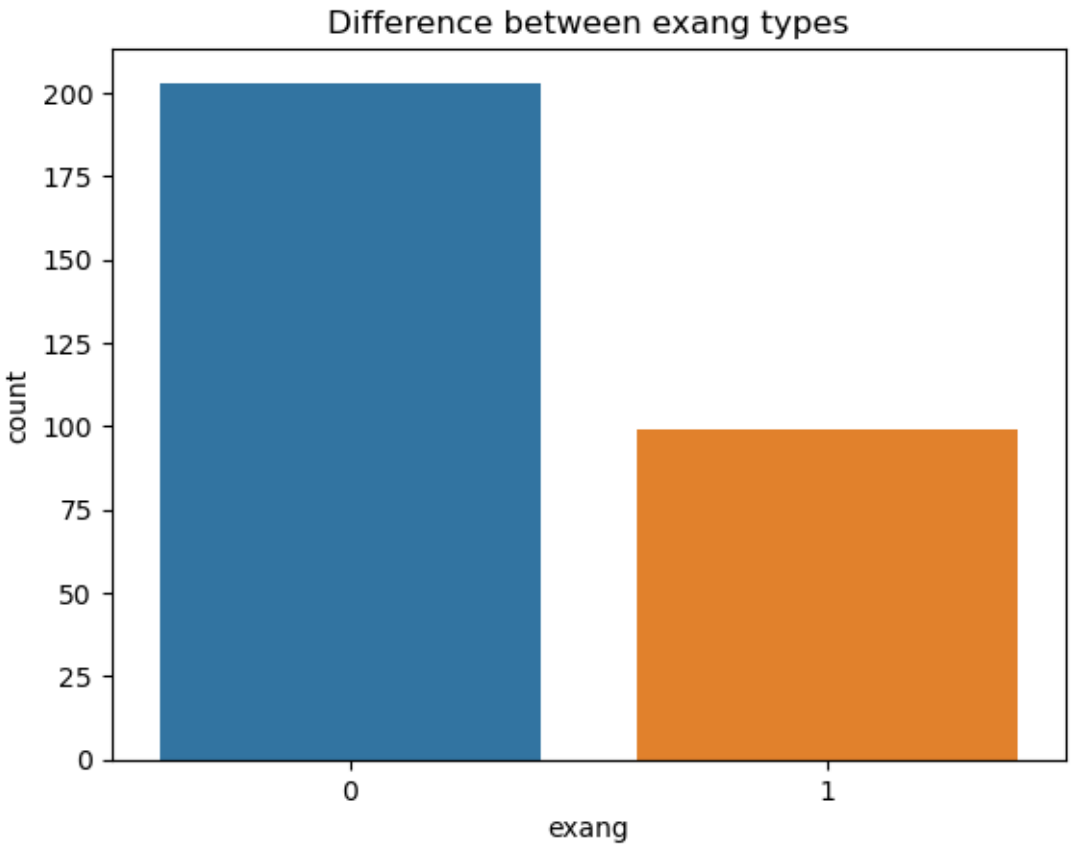
```
sns.countplot(data=df, x="restecg")
plt.title('Difference between restecg types')
plt.show()

sns.countplot(data=df, x="exang")
plt.title('Difference between exang types')
plt.show()

sns.countplot(data=df, x="slope")
plt.title('Difference between slope types')
plt.show()

sns.barplot(data=df, x="ca", y='target')
plt.title('Difference between ca types')
plt.show()

sns.countplot(data=df, x="thal")
plt.title('Difference between thal types')
plt.show()
```

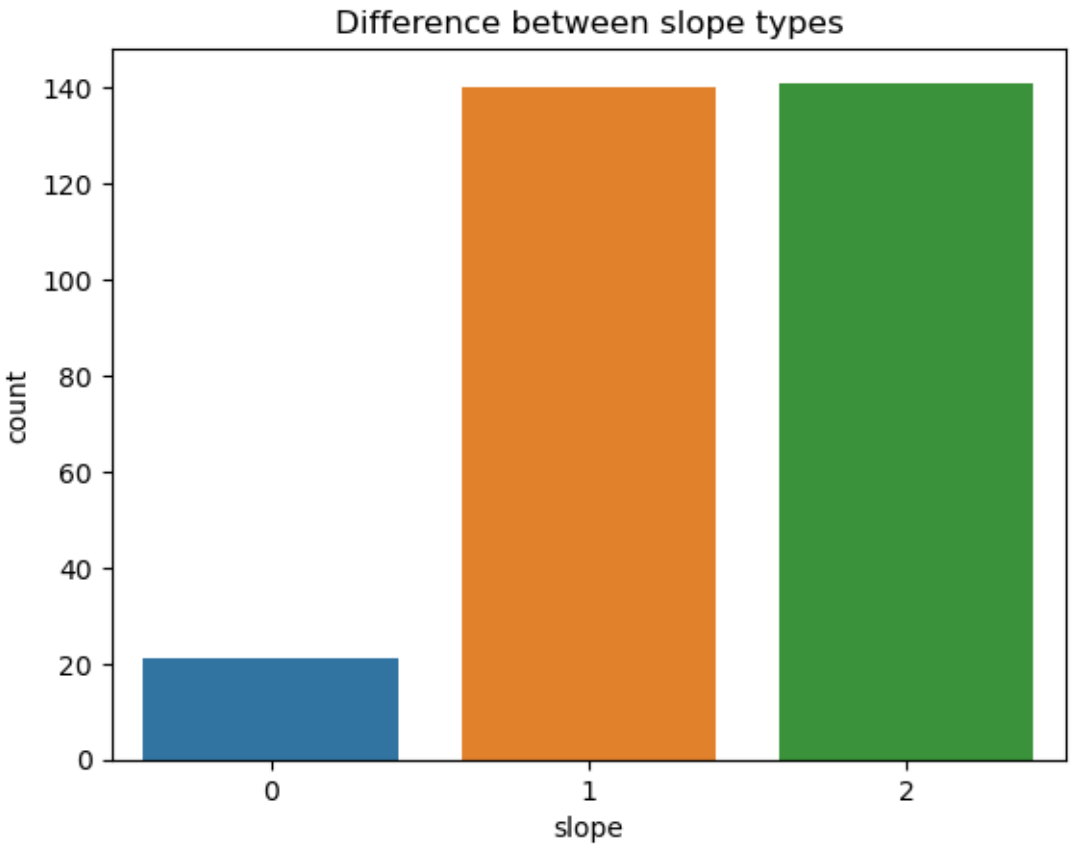


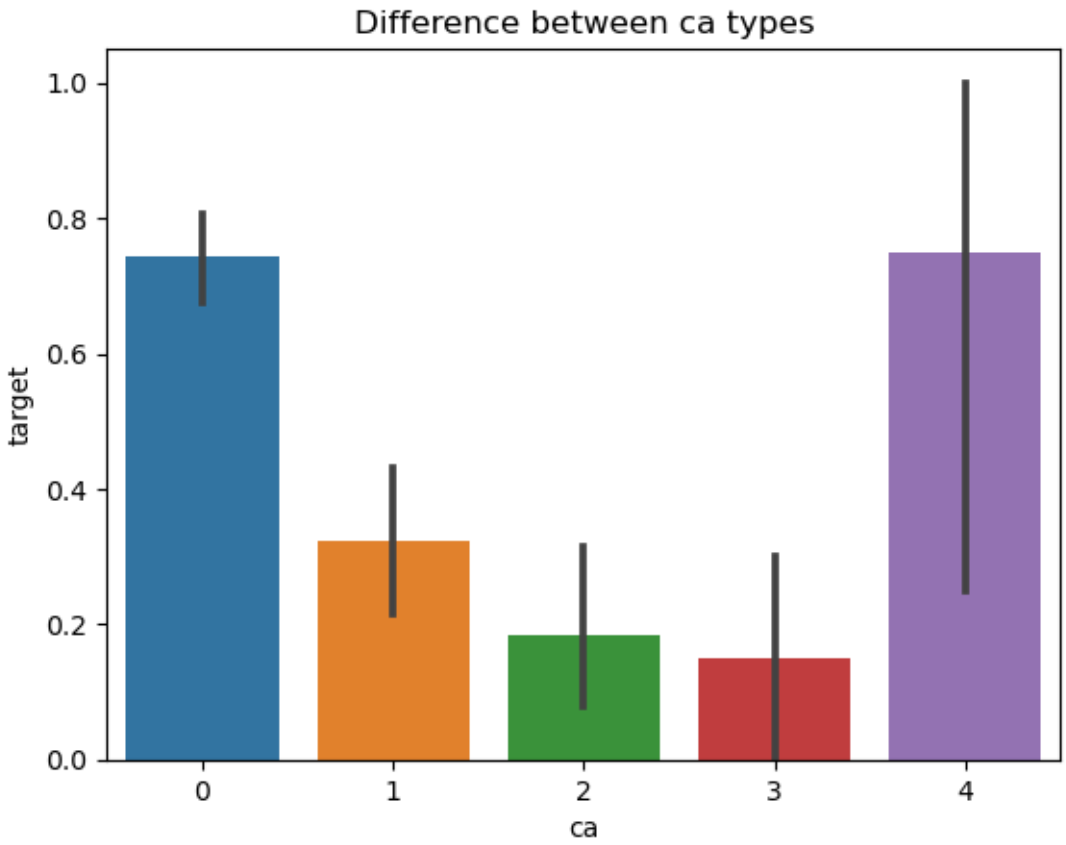

Difference between chest pain types
count
cp
0
1
2
3
0
20
40
60
80
100
120
140

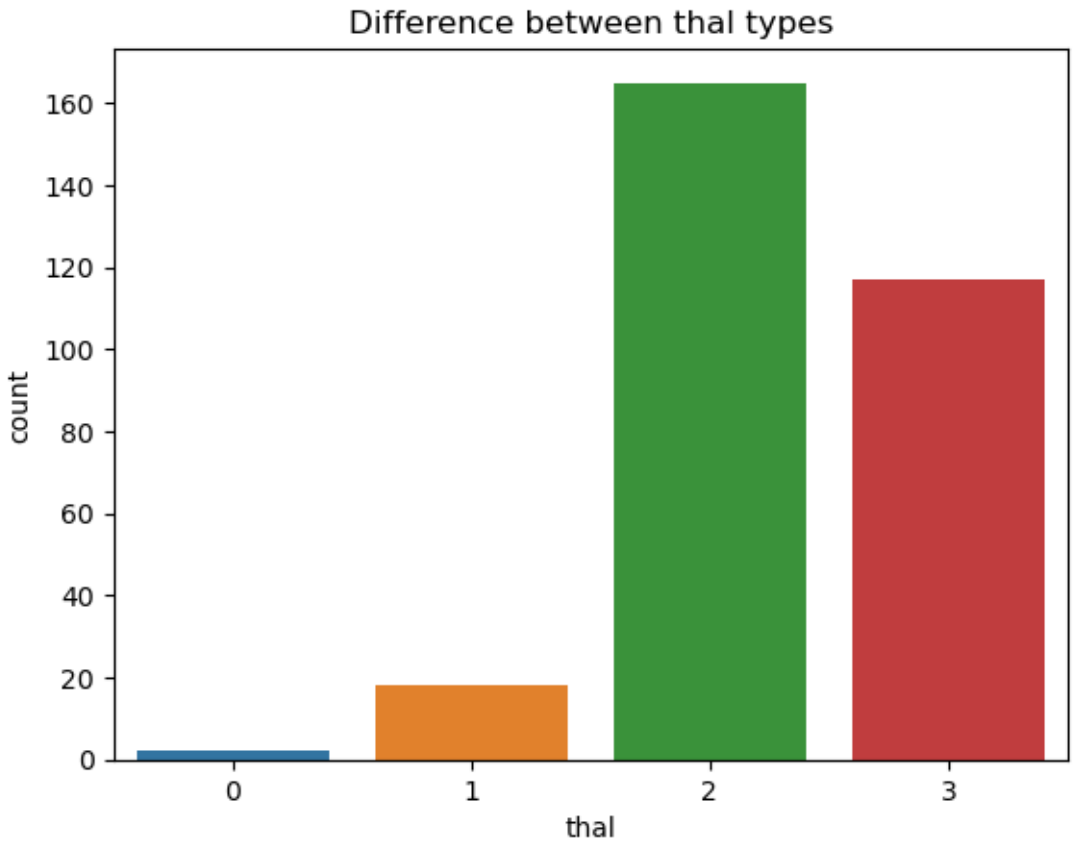

Difference between fbs types
250
200
150
100
50
0
count
0
1
fbs


Difference between restecg types
count
0
20
40
60
80
100
120
140
0
1
2
restecg


Difference between exang types
count
200
175
150
125
100
75
50
25
0
0
1
exang


Difference between slope types
140
120
100
80
60
40
20
0
count
0
1
2
slope


Difference between ca types
target
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
ca



[82]:
```
Age = df.loc[df['target']==1]
age0 = df.loc[df['target']==0]
Age
```

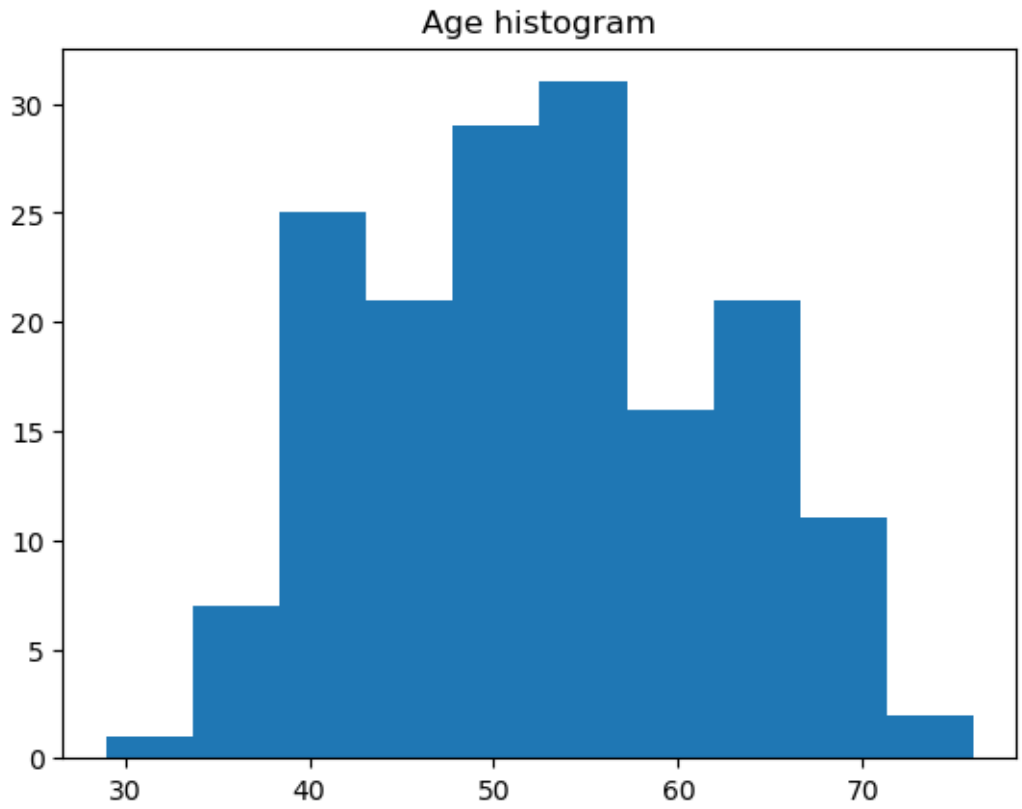
[82]:

	age	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	
3	56	1	1	120	236	0	1	178	0	0.8	
4	57	0	0	120	354	0	1	163	1	0.6	
..	…	…	..	…	…	…	…	…	…	…	
159	56	1	1	130	221	0	0	163	0	0.0	
160	56	1	1	120	240	0	1	169	0	0.0	
161	55	0	1	132	342	0	1	166	0	1.2	
162	41	1	1	120	157	0	1	182	0	0.0	
163	38	1	2	138	175	0	1	173	0	0.0	

	slope	ca	thal	target
0	0	0	1	1
1	0	0	2	1

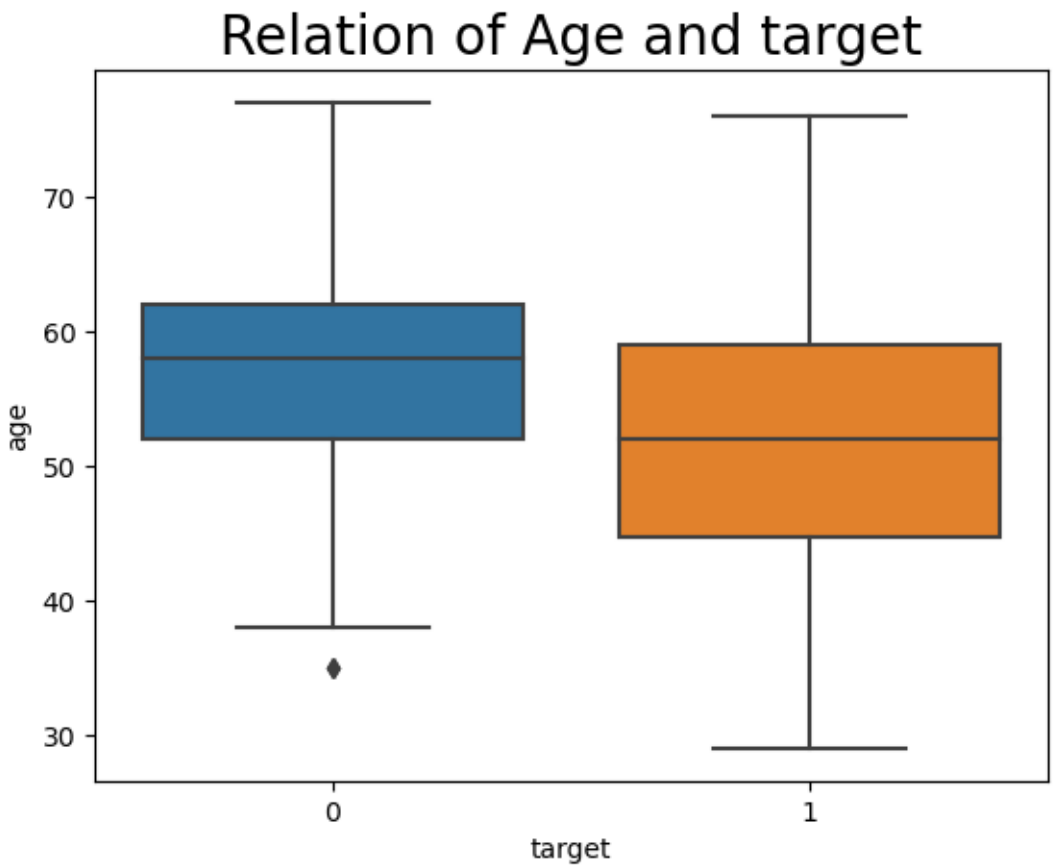
```
2         2   0     2        1
3         2   0     2        1
4         2   0     2        1
..      …  ..   …      …
159       2   0     3        1
160       0   0     2        1
161       2   0     2        1
162       2   0     2        1
163       2   4     2        1

[164 rows x 14 columns]
```

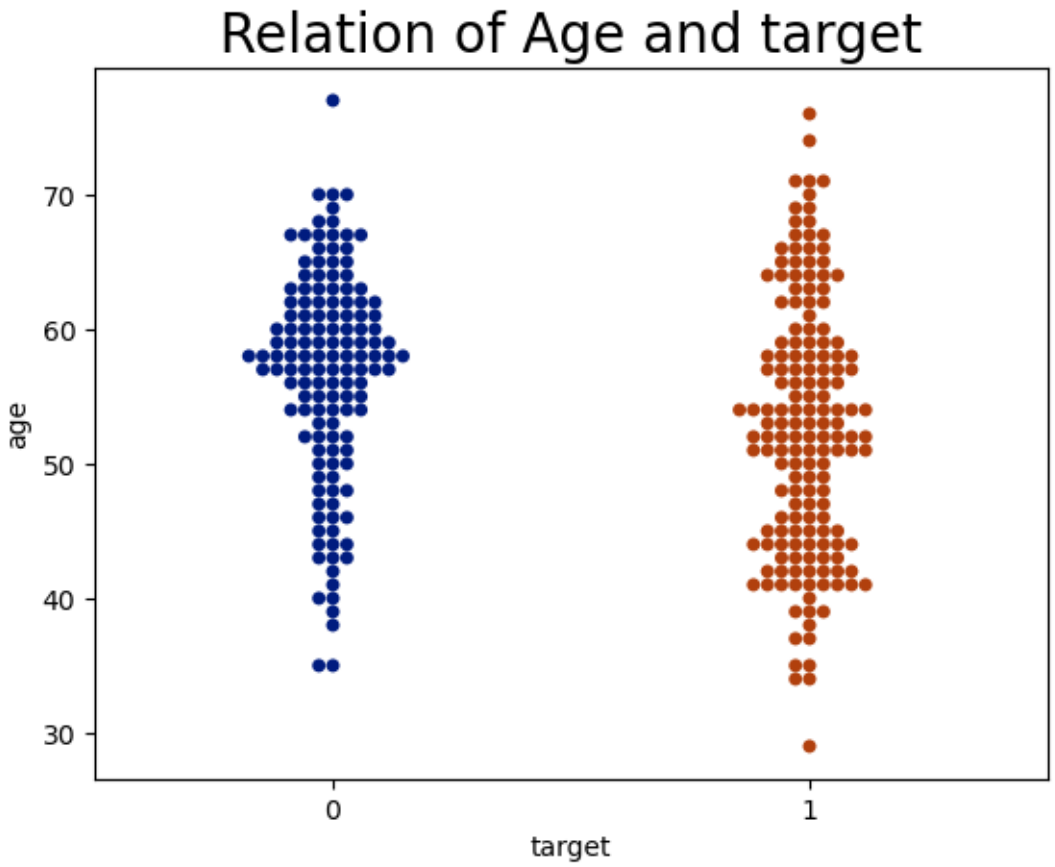
[343]:
```
plt.hist(x=Age['age'])
plt.title('Age histogram')
plt.show()
```



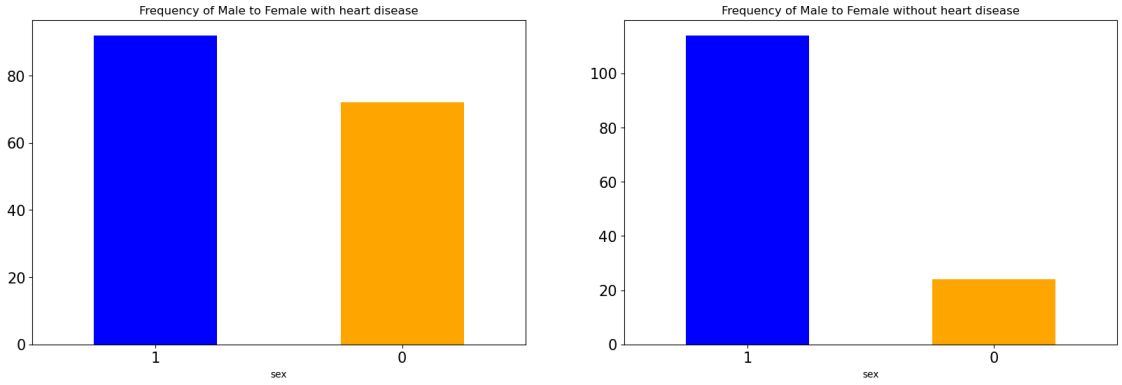
[111]:
```
# relation between age and target

sns.boxplot(data = df, x = 'target', y = 'age')
plt.title('Relation of Age and target', fontsize = 20)
plt.show()
```

```
sns.swarmplot(data = df, x = 'target', y = 'age', palette = 'dark')
plt.title('Relation of Age and target', fontsize = 20)
plt.show()
```

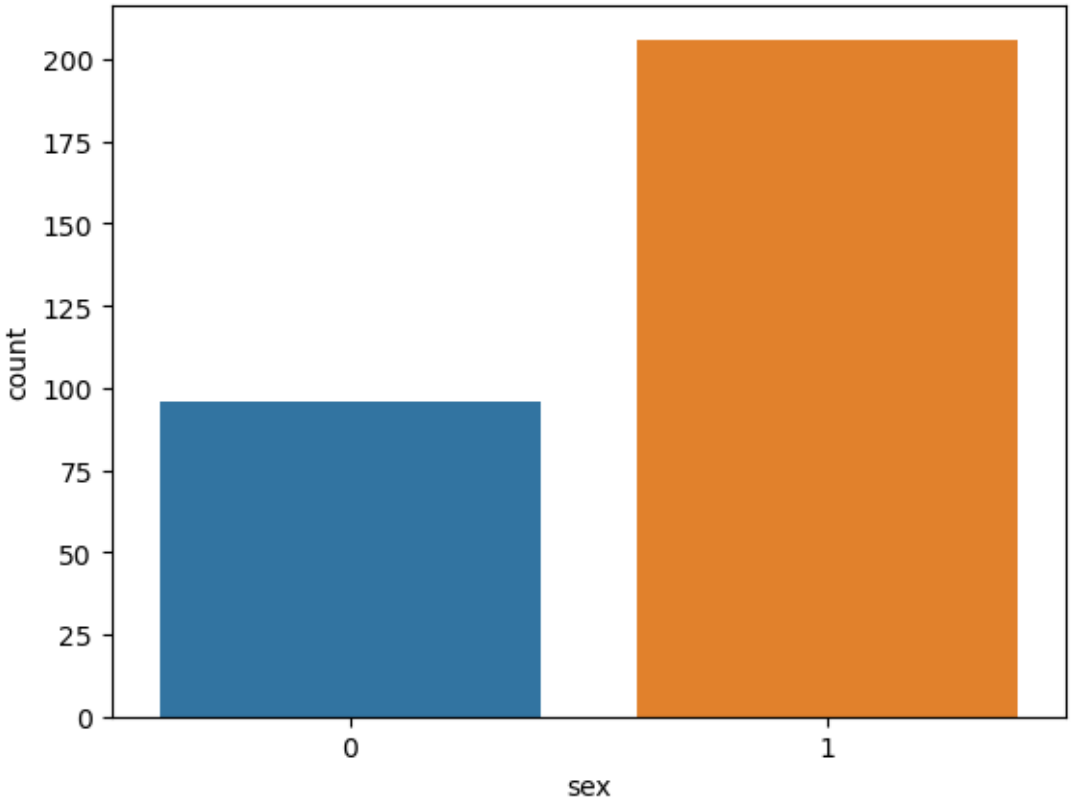




[432]:
```
fig, axarr = plt.subplots(1, 2, figsize=(20, 6))
cvd = df[df['target']==1].value_counts('sex').plot.bar(
    rot=0,
    fontsize= 15,
    title='Frequency of Male to Female with heart disease',
    color = ['blue', 'orange'],
    ax = axarr[0])

cvd1 = df[df['target']==0].value_counts('sex').plot.bar(
    rot=0,
    fontsize= 15,
    title='Frequency of Male to Female without heart disease',
    color = ['blue', 'orange'],
    ax = axarr[1])
plt.show()
sns.countplot(data=df, x ='sex')
```



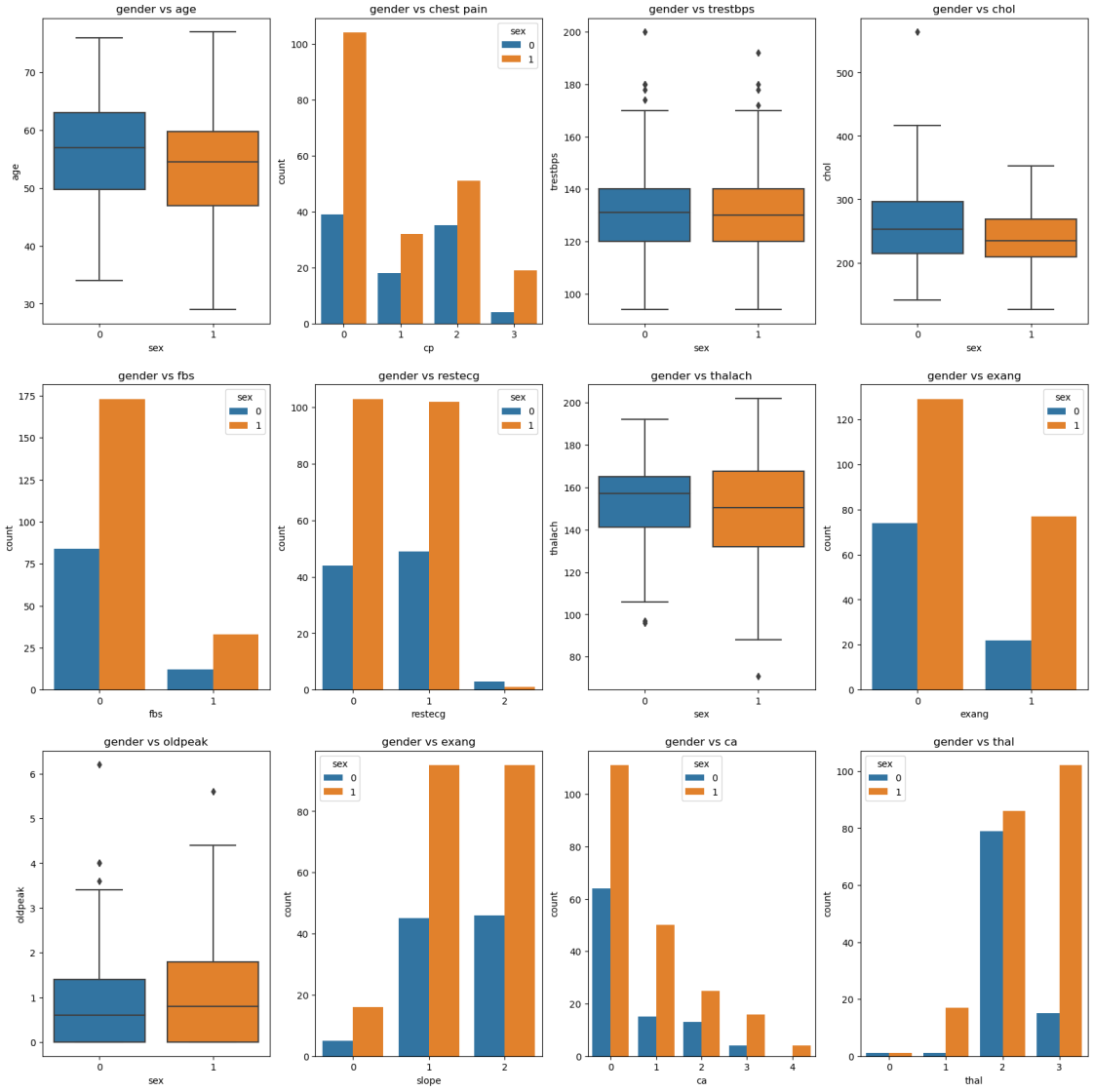
[432]: <AxesSubplot:xlabel='sex', ylabel='count'>



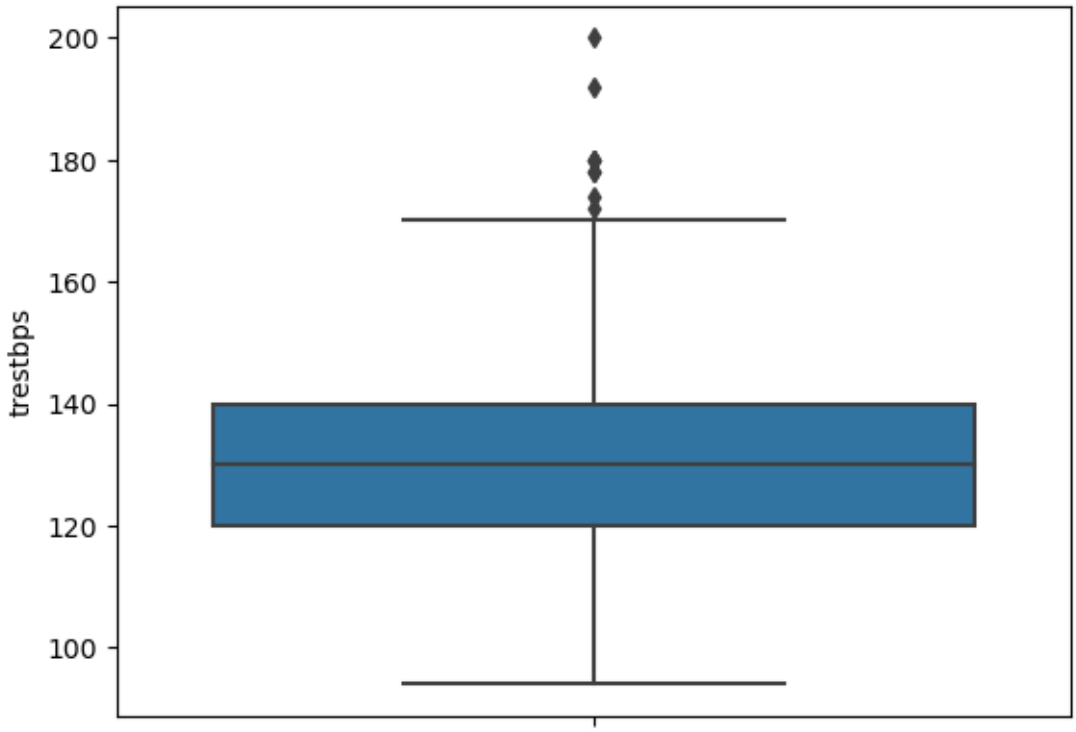
[214]:
```
fig, axes = plt.subplots(3, 4, figsize=(20, 20))
sns.boxplot(ax=axes[0][0] ,data = df, x = 'sex', y = 'age')
axes[0][0].set_title("gender vs age")

sns.countplot(ax = axes[0][1], data = df, x = 'cp', hue = 'sex')
```

```
axes[0][1].set_title("gender vs chest pain")

sns.boxplot(ax=axes[0][2], data = df, x = 'sex', y='trestbps')
axes[0][2].set_title('gender vs trestbps')

sns.boxplot(ax=axes[0][3], data = df, x = 'sex', y='chol')
axes[0][3].set_title('gender vs chol')

sns.countplot(ax = axes[1][0], data = df, x = 'fbs', hue = 'sex')
axes[1][0].set_title("gender vs fbs")

sns.countplot(ax = axes[1][1], data = df, x = 'restecg', hue = 'sex')
axes[1][1].set_title("gender vs restecg")

sns.boxplot(ax=axes[1][2], data = df, x = 'sex', y='thalach')
axes[1][2].set_title('gender vs thalach')

sns.countplot(ax = axes[1][3], data = df, x = 'exang', hue = 'sex')
axes[1][3].set_title("gender vs exang")

sns.boxplot(ax=axes[2][0], data = df, x = 'sex', y='oldpeak')
axes[2][0].set_title('gender vs oldpeak')

sns.countplot(ax = axes[2][1], data = df, x = 'slope', hue = 'sex')
axes[2][1].set_title("gender vs exang")

sns.countplot(ax = axes[2][2], data = df, x = 'ca', hue = 'sex')
axes[2][2].set_title("gender vs ca")

sns.countplot(ax = axes[2][3], data = df, x = 'thal', hue = 'sex')
axes[2][3].set_title("gender vs thal")
```

[214]: Text(0.5, 1.0, 'gender vs thal')



[439]:
```
sns.boxplot(data = df, y= 'trestbps')
df['trestbps'].describe()
rbp = df.loc[df['trestbps']>=180]
```



[440]:
```
rbp
```

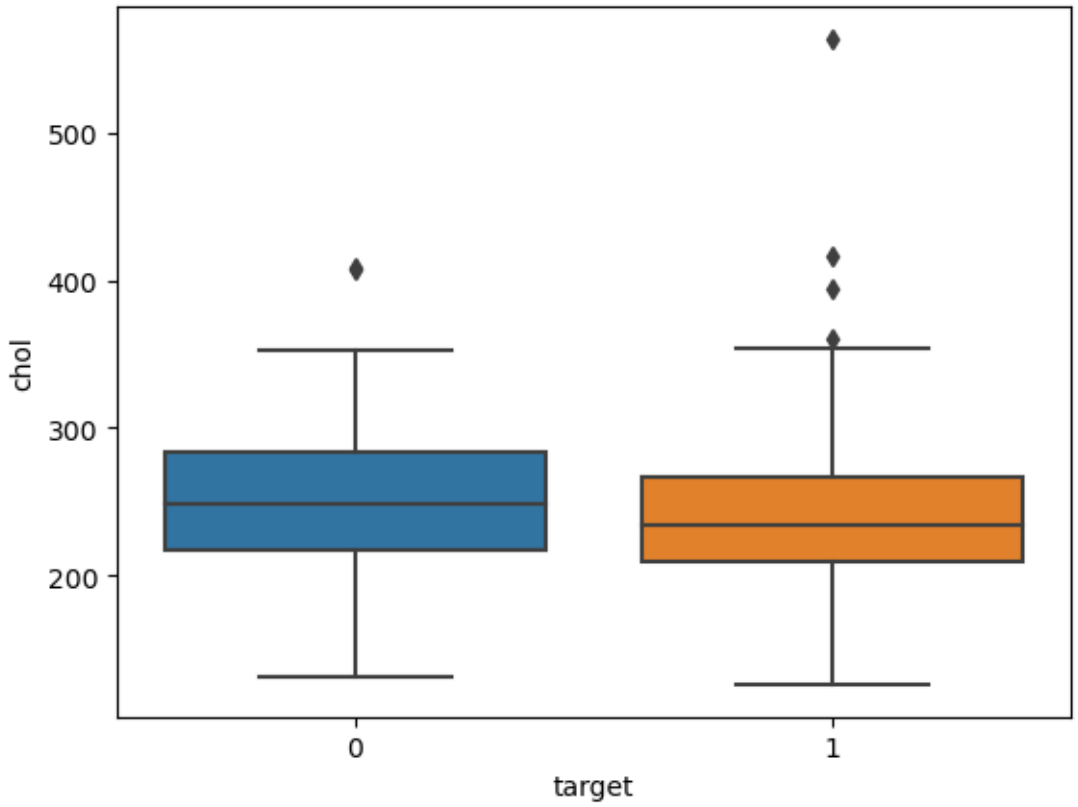
[440]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
110   64    0   0       180   325    0        1      154      1      0.0
203   68    1   2       180   274    1        0      150      1      1.6
223   56    0   0       200   288    1        0      133      1      4.0
248   54    1   1       192   283    0        0      195      0      0.0
266   55    0   0       180   327    0        2      117      1      3.4

     slope  ca  thal  target
110      2   0     2       1
203      1   0     3       0
223      0   2     3       0
248      2   1     3       0
266      1   0     2       0
```

[266]:
```
chol = df[['chol', 'target']]
chol.corr()
```

[266]:
```
              chol    target
chol      1.000000 -0.081437
target   -0.081437  1.000000
```

[268]:
```
sns.boxplot(data = chol, x = 'target', y = 'chol')
```

[268]:
```
<AxesSubplot:xlabel='target', ylabel='chol'>
```



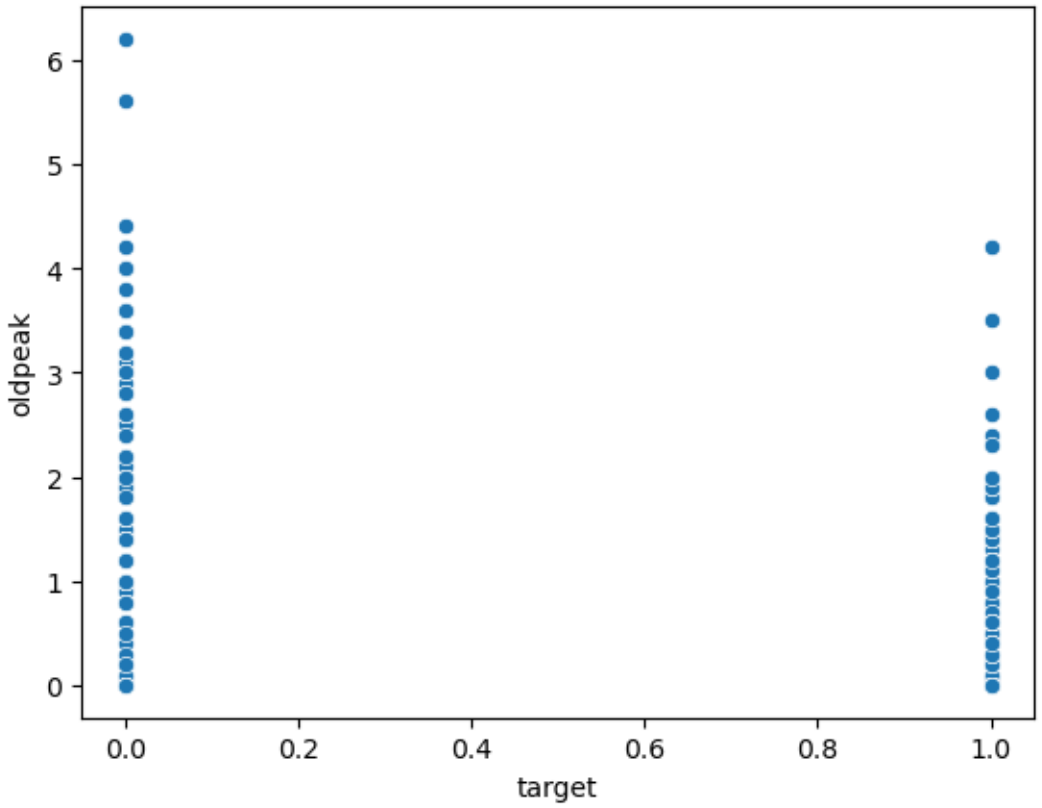
[269]:
```
chol = df[['oldpeak', 'target']]
chol.corr()
```

[269]:

	oldpeak	target
oldpeak	1.000000	-0.429146
target	-0.429146	1.000000

[271]:
```
sns.scatterplot(data = chol, x = 'target', y = 'oldpeak')
```

[271]:
```
<AxesSubplot:xlabel='target', ylabel='oldpeak'>
```



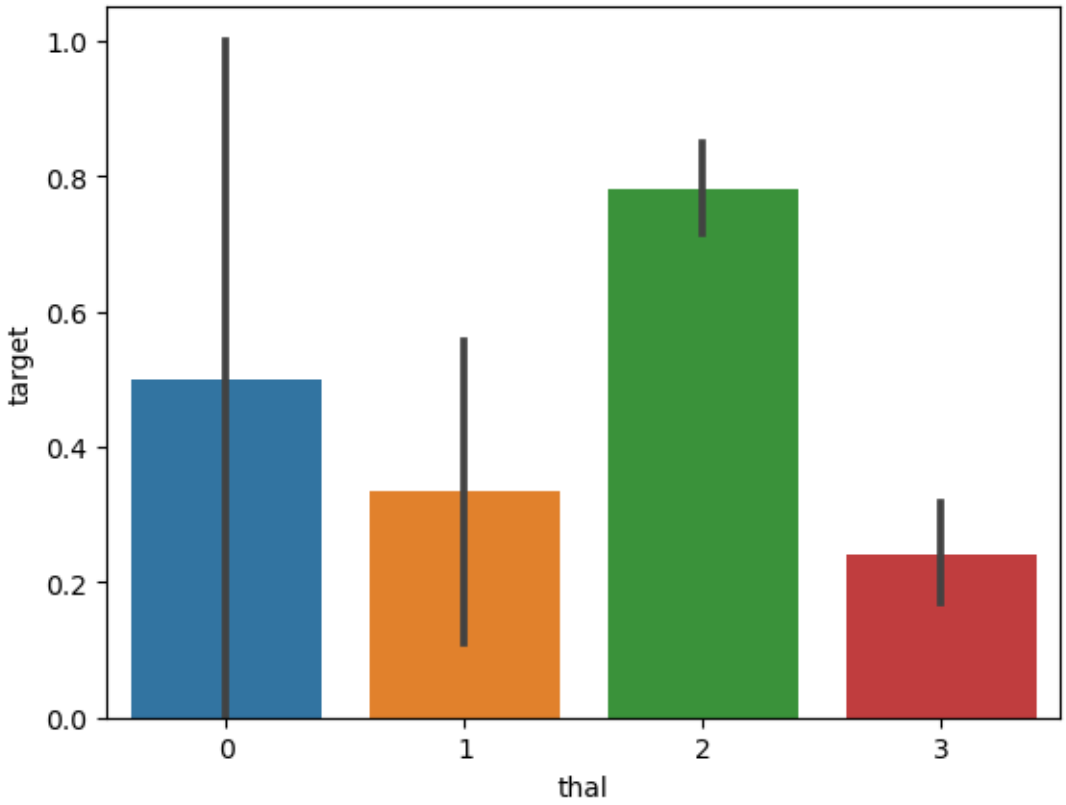
[273]:
```
thal = df[['thal', 'target']]
thal.corr()
```

[273]:

	thal	target
thal	1.000000	-0.343101
target	-0.343101	1.000000

[444]:
```
sns.barplot(data = df, x = 'thal', y = 'target')
```

[444]: <AxesSubplot:xlabel='thal', ylabel='target'>



[276]:
```
sns.pairplot(df)
```

[276]: <seaborn.axisgrid.PairGrid at 0x7f8c94e7d1c0>

[286]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

[287]:
```
```

[287]:

	age	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	
3	56	1	1	120	236	0	1	178	0	0.8	
4	57	0	0	120	354	0	1	163	1	0.6	
..	…	…	..	…	…	…	…	…	…	…	
298	57	0	0	140	241	0	1	123	1	0.2	

```
299   45    1   3       110   264    0       1       132      0      1.2
300   68    1   0       144   193    1       1       141      0      3.4
301   57    1   0       130   131    0       1       115      1      1.2
302   57    0   1       130   236    0       0       174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     …  ..   …
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[302 rows x 13 columns]
```

[368]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)
quantile_transformer = preprocessing.QuantileTransformer(random_state=0)
X_train_trans = quantile_transformer.fit_transform(X_train)
X_test_trans = quantile_transformer.transform(X_test)
```

```
/Users/lakshmimukkamala/opt/anaconda3/lib/python3.9/site-
packages/sklearn/preprocessing/_data.py:2590: UserWarning: n_quantiles (1000) is
greater than the total number of samples (241). n_quantiles is set to n_samples.
  warnings.warn(
```

[369]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train_trans, y_train)
```

[369]:
```
LogisticRegression(random_state=0)
```

[370]:
```
y_pred = classifier.predict(X_test_trans)
```

[371]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[23  4]
 [ 3 31]]
```

[371]: 0.8852459016393442

[395]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',␣
 ↪random_state = 0)
classifier.fit(X_train_trans, y_train)
```

[395]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

[396]:
```
y_pred = classifier.predict(X_test_trans)
```

[397]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[26  1]
 [ 7 27]]
```

[397]: 0.8688524590163934

[398]:
```
import statsmodels.api as sm
```

[399]:
```
results = sm.OLS(y, X).fit()
print(results.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                  target   R-squared (uncentered):
0.774
Model:                             OLS   Adj. R-squared (uncentered):
0.764
Method:                  Least Squares   F-statistic:
76.22
Date:                 Wed, 08 Feb 2023   Prob (F-statistic):
3.05e-85
Time:                         21:43:38   Log-Likelihood:
-111.63
No. Observations:                 302   AIC:
249.3
Df Residuals:                     289   BIC:
297.5
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
```

```
age            0.0035      0.002      1.503      0.134      -0.001       0.008
sex           -0.1706      0.047     -3.652      0.000      -0.263      -0.079
cp             0.1091      0.023      4.812      0.000       0.064       0.154
trestbps      -0.0008      0.001     -0.708      0.480      -0.003       0.001
chol          -0.0001      0.000     -0.254      0.799      -0.001       0.001
fbs            0.0084      0.060      0.139      0.890      -0.110       0.127
restecg        0.0686      0.040      1.728      0.085      -0.010       0.147
thalach        0.0050      0.001      5.605      0.000       0.003       0.007
exang         -0.1202      0.051     -2.350      0.019      -0.221      -0.020
oldpeak       -0.0526      0.023     -2.274      0.024      -0.098      -0.007
slope          0.0887      0.043      2.078      0.039       0.005       0.173
ca            -0.1120      0.023     -4.924      0.000      -0.157      -0.067
thal          -0.1021      0.036     -2.866      0.004      -0.172      -0.032
==============================================================================
Omnibus:                        7.900   Durbin-Watson:                   1.048
Prob(Omnibus):                  0.019   Jarque-Bera (JB):                8.133
Skew:                          -0.401   Prob(JB):                       0.0171
Kurtosis:                       2.935   Cond. No.                         962.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[400]:
```
no = df[['age', 'trestbps', 'chol', 'fbs']]
```

[401]:
```
yes = df[['sex', 'cp', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca',␣
 ↪'age']]
```

[403]:
```
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
X_train, X_test, y_train, y_test = train_test_split(yes, y, test_size = 0.2,␣
 ↪random_state = 0)
quantile_transformer = preprocessing.QuantileTransformer(random_state=0)
X_train_trans = quantile_transformer.fit_transform(X_train)
X_test_trans = quantile_transformer.transform(X_test)
```

```
/Users/lakshmimukkamala/opt/anaconda3/lib/python3.9/site-
packages/sklearn/preprocessing/_data.py:2590: UserWarning: n_quantiles (1000) is
greater than the total number of samples (241). n_quantiles is set to n_samples.
  warnings.warn(
```

[410]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
y_pred = classifier.predict(X_test)
```

```
/Users/lakshmimukkamala/opt/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:814: 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(
```

[411]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[22  5]
 [ 4 30]]
```

[411]:
```
0.8524590163934426
```

[428]:
```
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',␣
↪random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

[429]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[23  4]
 [ 7 27]]
```

[429]:
```
0.819672131147541
```

[ ]:
```

```