Exploratory Data Analysis on House Sale Prices Data

This report contains the process of transforming of raw data to a predictive model data, including data processing, cleaning, feature engineering, exploratory data analysis, hypothesis test.

1. Dataset and attributes

Before we can begin any analysis, we first need to acquire the dataset. For this, "heart.csv" was downloaded from Kaggle-heart and saved it into the os folder. This dataset contains the age, chol, ca, cp etc. To analyze in detail, first, the necessary libraries such as Pandas to store and transform the data; Numpy to perform scientific computing or mathematical operations on data set; and Matplotlib and Seaborn for the data visualizations are imported in Python and then the our dataset is placed into a Pandas data frame by defining correct path as shown below:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
%matplotlib inline
```

Let's load the dataset

In [2]:
```
df = pd.read_csv("C://Users//anjaneya//Documents//Path to Data Scientist//Course 1 - Exploratory Data Analysis//Lab 4//heart.csv")
df.head()
```

Out[2]:

	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

Features in the dataset

- **age** : Indicates the age of a person
- **sex** : Indicates whether a person is male or female (1 = male, 0 = female)
- **cp**: The chest pain experienced (Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic)
- **trestbps**: The person's resting blood pressure (mm Hg on admission to the hospital)
- **chol**: The person's cholesterol measurement in mg/dl
- **fbs**: The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)
- **restecg**: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)
- **thalach**: The person's maximum heart rate achieved
- **exang**: Exercise induced angina (1 = yes; 0 = no)
- **oldpeak**: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot)
- **slope**: the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)
- **ca**: The number of major vessels (0-3)
- **thal**: A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)
- **target**: Heart disease (0 = no, 1 = yes)

Let's get general information about our dataset

In [3]:
```
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
```

Let's know the general information like it's shape

In [5]:
```
print("The shape of dataframe is ",df.shape)
print("\nTotal various data types ",df.dtypes.value_counts())
```

```
The shape of dataframe is  (303, 14)

Total various data types  int64      13
float64     1
dtype: int64
```

One can see clearly that the dataset contains total 14 variables (columns) and 303 observations (rows). From 14 variables, first 13 variables are features variables and last column ‘target’ is the target variable. Moreover, the dataset contains the data of various data types:integers and floats.

Let's find if there are any null values in our dataset

In [6]:
```
df.isna().sum(axis=0)
```

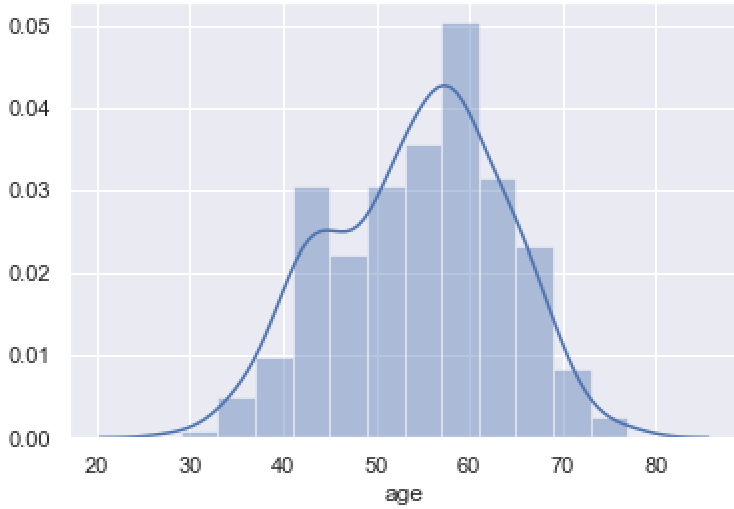
Out[6]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

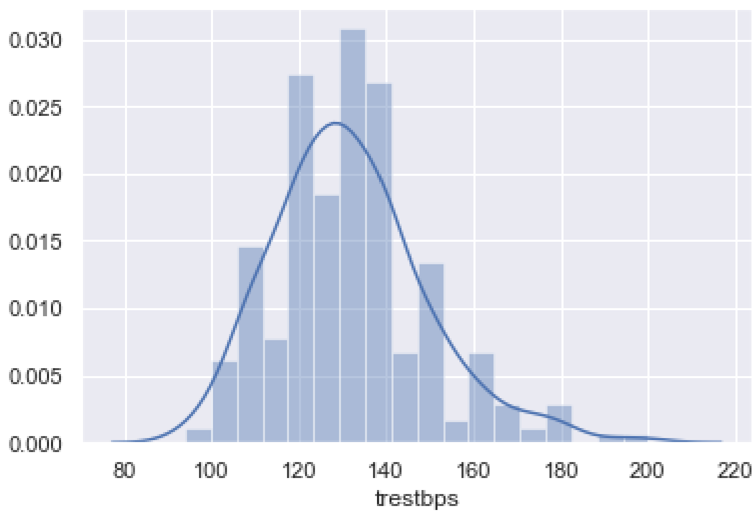
2. Data cleaning and Set label and features

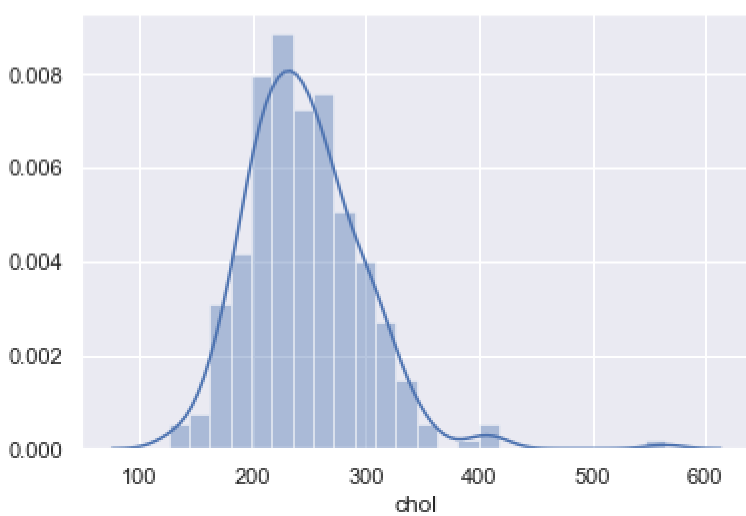
In [13]:
```
sns.set()
sns.set_context("notebook")
featuresDf = df.drop(columns="target")
features = featuresDf.columns.values
labelDf = df["target"]
label = "target"
```

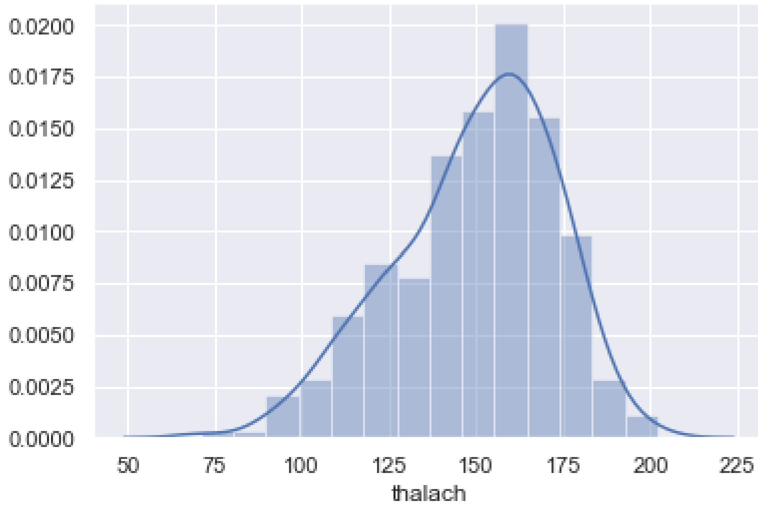
Check the distribution of the non-categorical features

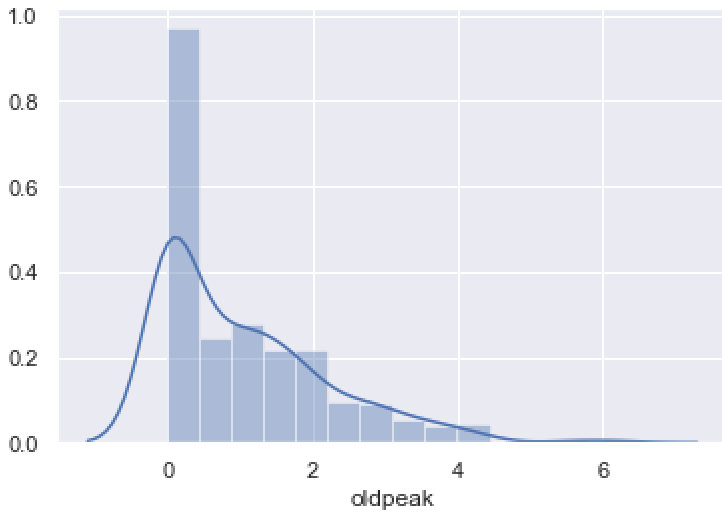
In [9]:
```
for feature in features:
    df[feature] = df[feature].astype(float)
    uniqueValueCount = len(df[feature].unique())
    if(uniqueValueCount> 4):
        sns.distplot(df[feature])
    plt.show()
```

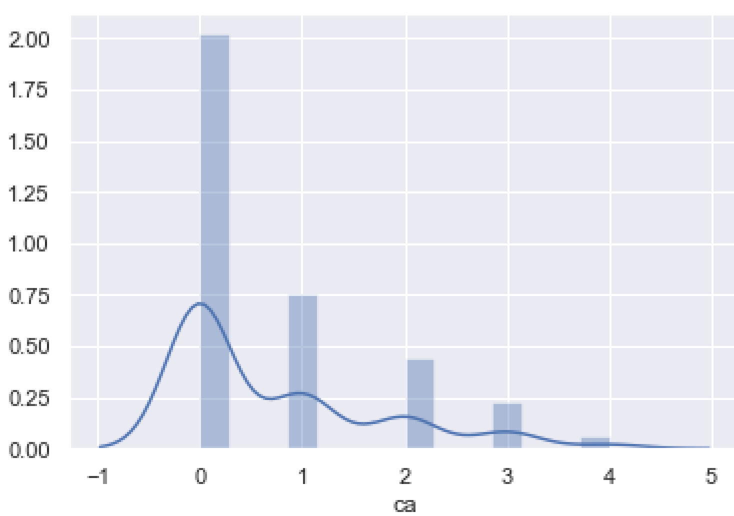

0.05
0.04
0.03
0.02
0.01
0.00
20
30
40
50
60
70
80
age


0.030
0.025
0.020
0.015
0.010
0.005
0.000
80
100
120
140
160
180
200
220
trestbps


0.008
0.006
0.004
0.002
0.000
100
200
300
400
500
600
chol



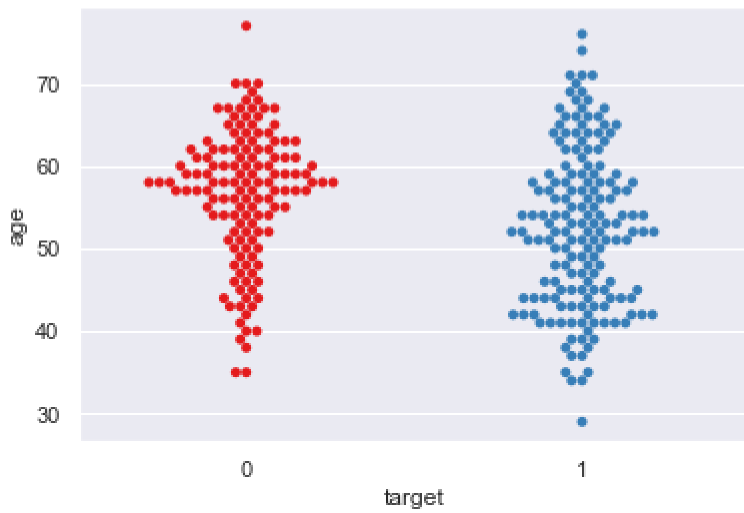


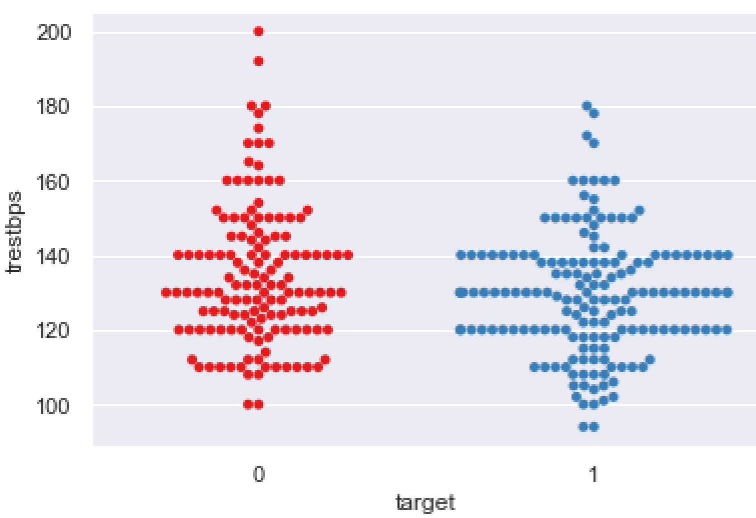


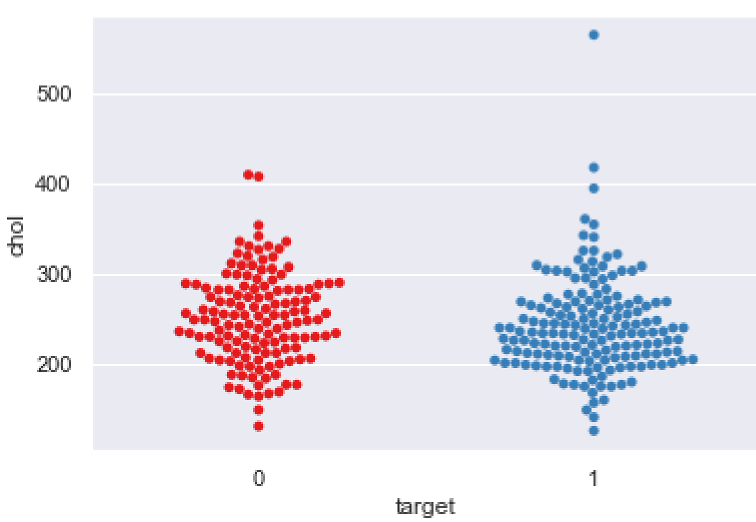
Scatter for all feature pair combinations

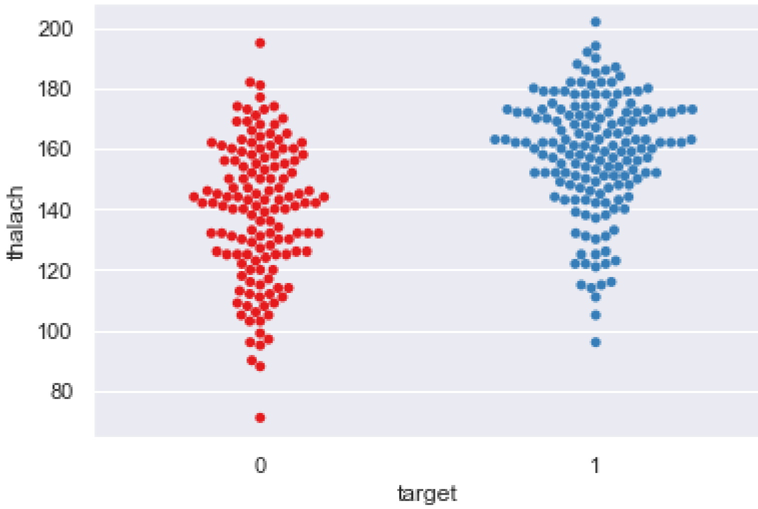
In [10]:
```
for feature in features:
    df[feature] = df[feature].astype(float)
    uniqueValueCount = len(df[feature].unique())
    if(uniqueValueCount> 4):
        sns.swarmplot(x=label,y=feature,data=df,palette="Set1", split=True)
    plt.show()
```

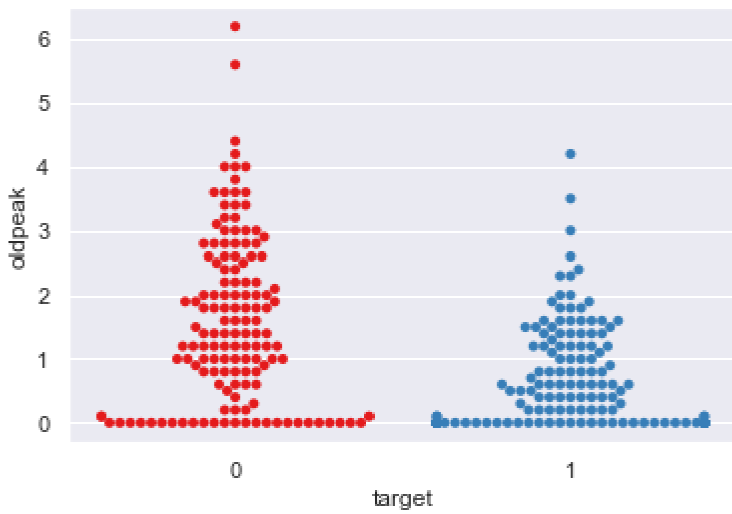
```
c:\users\anjaneya\appdata\local\programs\python\python38\lib\site-packages\se
aborn\categorical.py:2971: UserWarning: The `split` parameter has been rename
d to `dodge`.
  warnings.warn(msg, UserWarning)
```

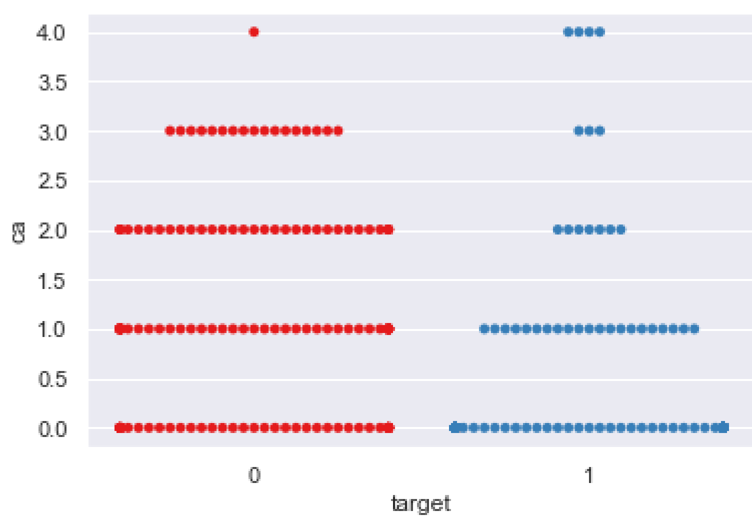






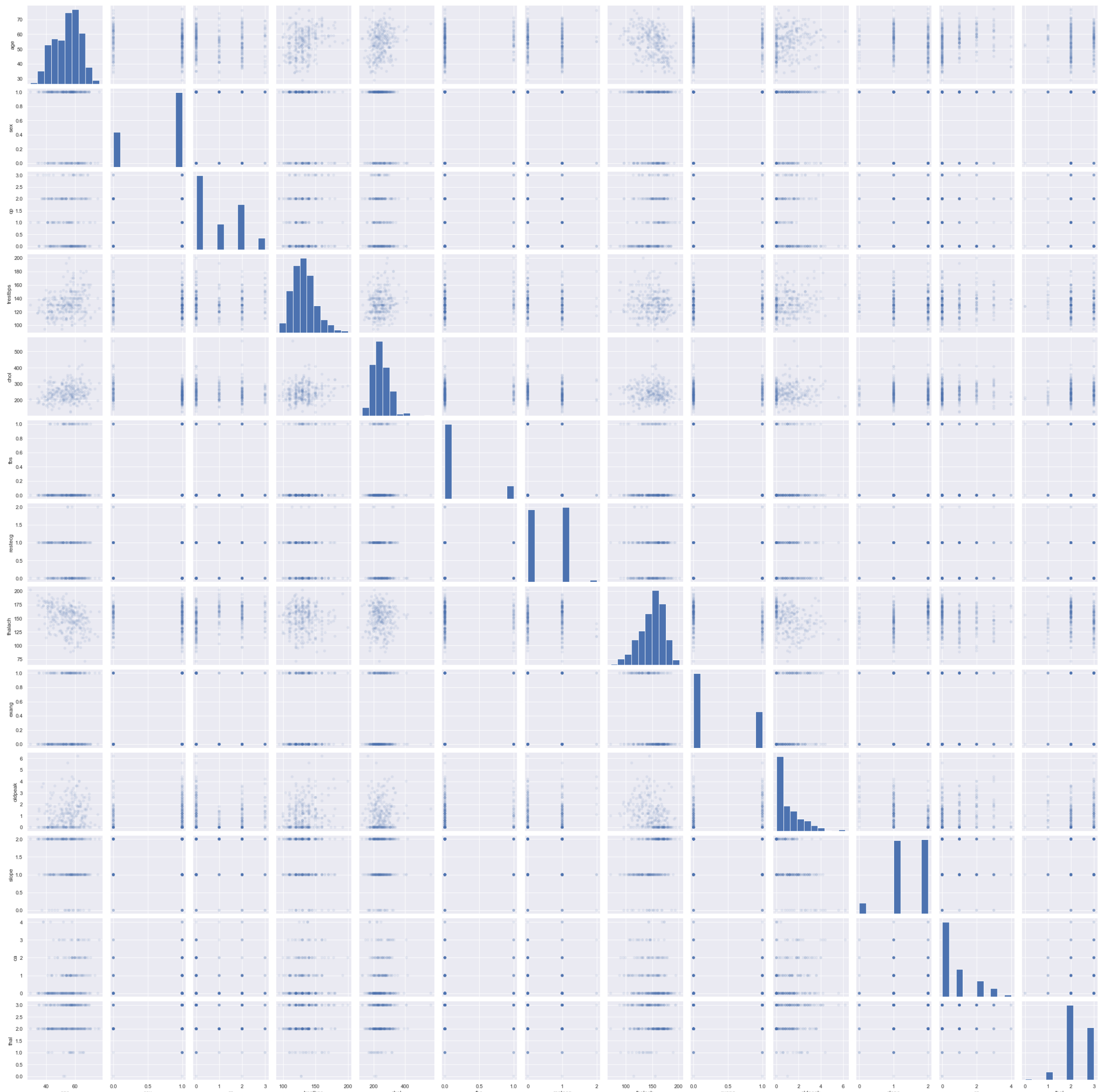

200
180
160
140
120
100
80
thalach
0
1
target


6
5
4
3
2
1
0
oldpeak
0
1
target


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ca
0
1
target

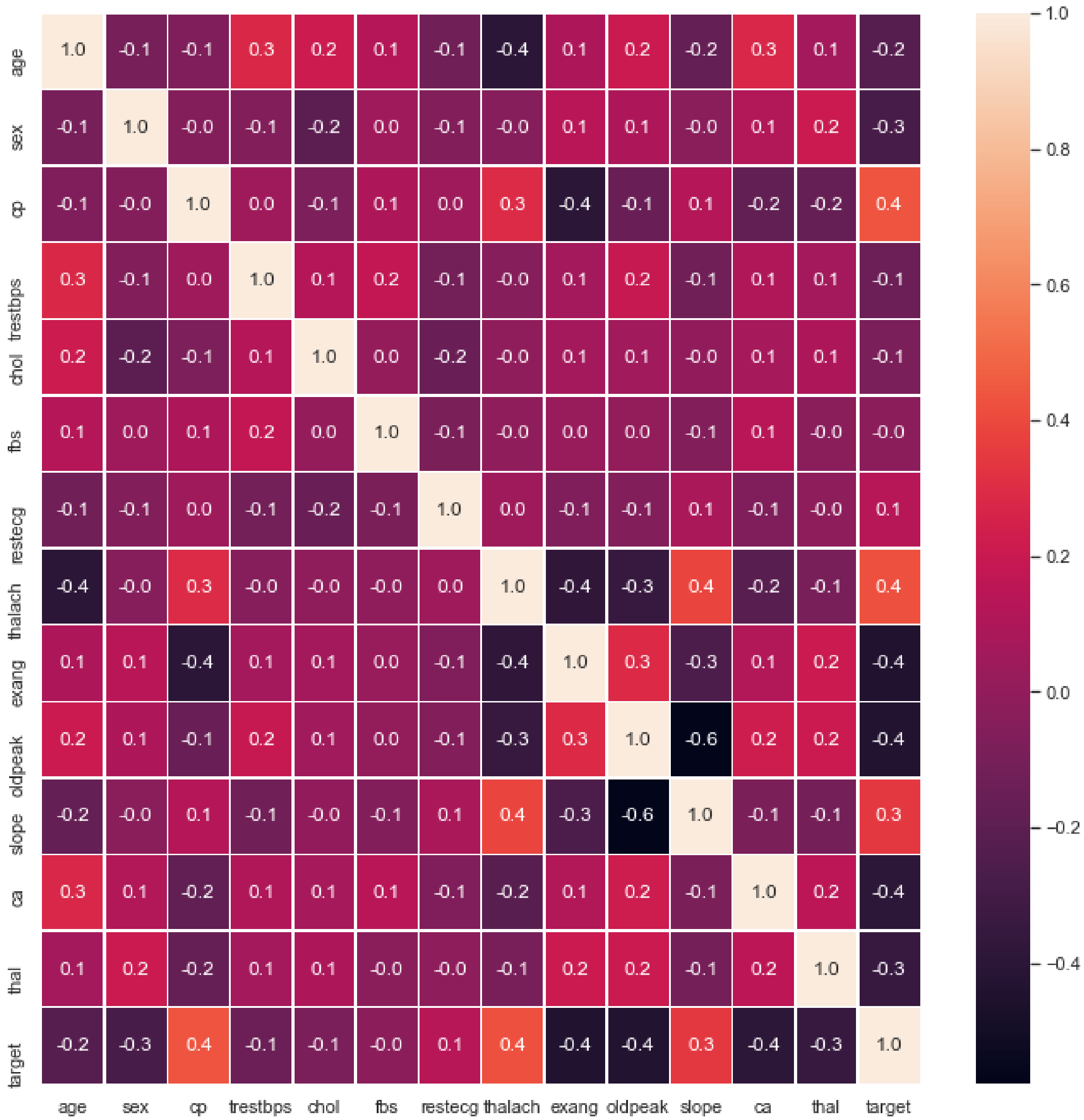
In [12]:
```
sns.pairplot(featuresDf,plot_kws=dict(alpha=.1, edgecolor='none'))
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x1b8ed790>



Correlation Map

In [11]:
```
f,ax = plt.subplots(figsize=(12,12))
sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```



3 Exploratory Data Analysis

3.1 Univariate Analysis

3.1.1 Numerical Variables

In future modeling, our predicting parameter is ‘SalePrice’. Hence, we start exploring the data with target. More information about ‘target’ feature is retrieved by using describe function as follows:

In [14]:
```
df.target.describe()
```

Out[14]:
```
count    303.000000
mean       0.544554
std        0.498835
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: target, dtype: float64
```

Count displayes the total numbers of rows in the series. We can see that the average target in our dataset is close to 0.544 with most of the values falling within the 0 to 1 range.

3.2 Bivariate Analysis

3.2.1 Numerical Variables

To perform bivariate analysis for numerical variables, all numerical features are filtered out as shown below:

In [15]:
```
num_features = df.select_dtypes(include=[np.number])
num_features.dtypes
```

Out[15]:
```
age         float64
sex         float64
cp          float64
trestbps    float64
chol        float64
fbs         float64
restecg     float64
thalach     float64
exang       float64
oldpeak     float64
slope       float64
ca          float64
thal        float64
target        int64
dtype: object
```

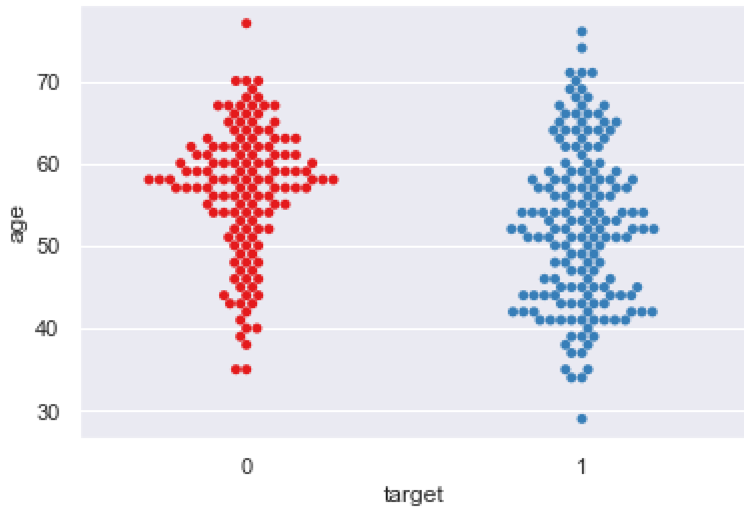
Then, the correlation method (.corr()) is used to get the relationship between the each feature. As ‘target’ is our target variable, we examine the correlations between all features and the ‘target’ as shown below:

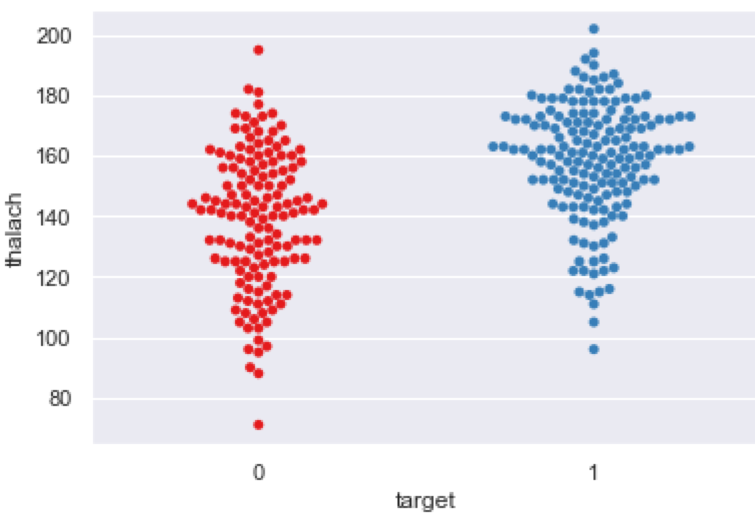
In [16]:
```
corr = num_features.corr()
print(corr['target'].sort_values(ascending = False)[:5],'\n')
print(corr['target'].sort_values(ascending = False)[-5:])
```

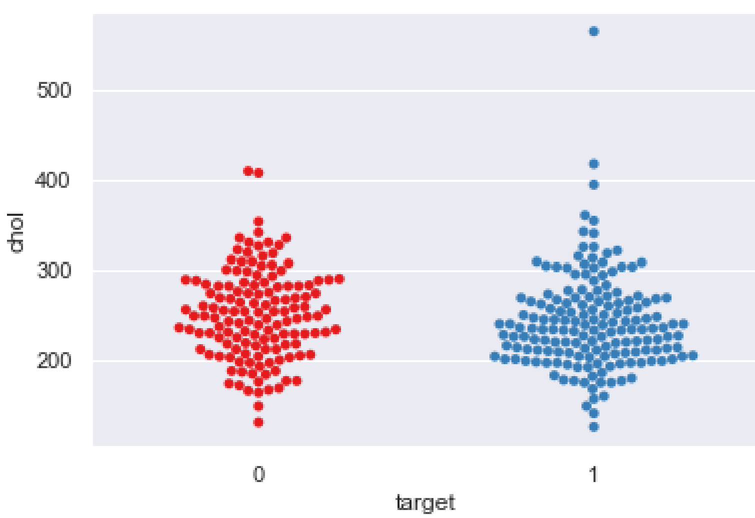
```
target      1.000000
cp          0.433798
thalach     0.421741
slope       0.345877
restecg     0.137230
Name: target, dtype: float64

sex        -0.280937
thal       -0.344029
ca         -0.391724
oldpeak    -0.430696
exang      -0.436757
Name: target, dtype: float64
```

Let's see correlated value distributions of both target = 1 and target = 0 to compare, using swarmplot

In [17]:
```
targetCorrelated = ["age","thalach","chol"]
for feature in targetCorrelated:
    sns.swarmplot(x=label,y=feature,data=df,palette="Set1", split=True)
    plt.show()
```







Lets focus on target = 1

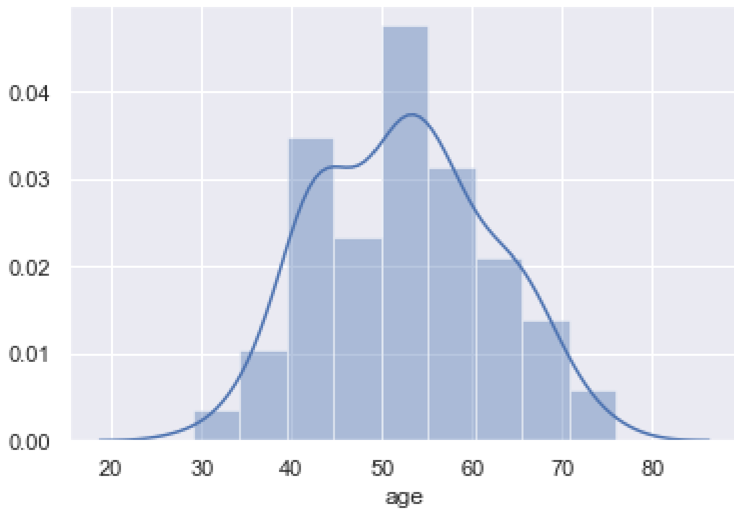
In [18]:
```
target1 = df.loc[df.target==1]
target0 = df.loc[df.target==0]
print(target1.head())
print(target0.head())
```

```
    age  sex   cp  trestbps   chol  fbs  restecg  thalach  exang  oldpeak  \
0  63.0  1.0  3.0     145.0  233.0  1.0      0.0    150.0    0.0      2.3   
1  37.0  1.0  2.0     130.0  250.0  0.0      1.0    187.0    0.0      3.5   
2  41.0  0.0  1.0     130.0  204.0  0.0      0.0    172.0    0.0      1.4   
3  56.0  1.0  1.0     120.0  236.0  0.0      1.0    178.0    0.0      0.8   
4  57.0  0.0  0.0     120.0  354.0  0.0      1.0    163.0    1.0      0.6   

   slope   ca  thal  target  
0    0.0  0.0   1.0       1  
1    0.0  0.0   2.0       1  
2    2.0  0.0   2.0       1  
3    2.0  0.0   2.0       1  
4    2.0  0.0   2.0       1  
      age  sex   cp  trestbps   chol  fbs  restecg  thalach  exang  oldpeak
\
165  67.0  1.0  0.0     160.0  286.0  0.0      0.0    108.0    1.0      1.5   
166  67.0  1.0  0.0     120.0  229.0  0.0      0.0    129.0    1.0      2.6   
167  62.0  0.0  0.0     140.0  268.0  0.0      0.0    160.0    0.0      3.6   
168  63.0  1.0  0.0     130.0  254.0  0.0      0.0    147.0    0.0      1.4   
169  53.0  1.0  0.0     140.0  203.0  1.0      0.0    155.0    1.0      3.1   

     slope   ca  thal  target  
165    1.0  3.0   2.0       0  
166    1.0  2.0   3.0       0  
167    0.0  2.0   2.0       0  
168    1.0  1.0   3.0       0  
169    0.0  0.0   3.0       0  
```

Age of target = 1

In [19]:
```
sns.distplot(target1.age)
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x22428df0>



Do ages of the people who have target = 1 have correlation with other features?

In [20]:
```
cor_mat = target1.corr()
print(cor_mat["age"])
```

```
age         1.000000
sex        -0.190231
cp          0.024934
trestbps    0.274698
chol        0.257154
fbs         0.155415
restecg    -0.084360
thalach    -0.525801
exang       0.046990
oldpeak     0.174594
slope      -0.109380
ca          0.117463
thal        0.080959
target           NaN
Name: age, dtype: float64
```

4 Hypothesis Formulation and Testing

4.1 Null and Alternative Hypothesis

4.1.1 Hypothesis-1

Null (H0): The earlier bult data of the house result in the lower house price. Alternative (H1): The earlier built date of the house does not result in the lower price.

4.1.2 Hypothesis-2

Null (H0): The higher Overall Quality of the house results int the higher house price. Alternative (H1): The higher Overall Quality of the house does not result in the higher price.

4.1.3 Hypothesis-3

Null (H0): The house with big living area is costly. Alternative (H1): The house with big living area is not costly.

4.2 Significance Test for Hypothesis-1

4.2.1 Hypothesis-1

Null (H0): The earlier bult data of the house result in the lower house price. Alternative (H1): The earlier built date of the house does not result in the lower house price.

4.2.2 Choose the Level of Significance

Set α to be 0.05 which is referred to 95% confidence level.

4.2.3 Determine the Probability

Usually, the probability or known as p-Value is used to decide whether to accept or reject the null hypothesis. If the p-value is less than or equal to the level of significance, the null hypothesis is rejected.

In [21]:
```
from scipy.stats import pearsonr
from scipy.stats import spearmanr
import math
from scipy.stats import ttest_ind
```

In [22]:
```
def MinMax(data,col):
    newCol = df[col].copy()
    newCol -= newCol.min()
    newCol /= data[col].max()
    return newCol
```

In [23]:
```
def correlationHypo(data,feature1, feature2,threshold=0.05):
    data1, data2 = df[feature1],df[feature2]
    stat0, p0 = pearsonr(data1,data2)
    stat, p = spearmanr(data1, data2)
    print('stat=%.3f, p=%.3f' % (stat, p))
    p = min(p,p0)
    if p > threshold:
        print('p value {:.3} greater than {} that means they are probably inde
pendent\nNull Hypothesis Accepted!'.format(p,threshold))
    else:
        print('p value {:.3} lower than {} that means they are probably depend
ent\nNull Hypothesis Rejected!'.format(p,threshold))
```

In [24]:
```
correlationHypo(df,"chol","thalach")
```

```
stat=-0.047, p=0.417
p value 0.417 greater than 0.05 that means they are probably independent
Null Hypothesis Accepted!
```

5 Suggestions for Further Analyzing the Data

Before running the machine learning algorithms, it is better to pay bit more time working with data, especially the feature engineering, which is the process of pre-processing data in a way that optimizes learning

Summary

The above dataset is downloaded from Kagle and doesn't conatin any missing data or strings. Missing data can be handled by taking the mean of the entire column (general practice) and strings using one-hot-encoding or get_dummies(). Of course more data is always useful to train and test the models. For our study this data is helpfl enough. So, finally we found whether our null hypothesis is accepted or not.

In [ ]:
```

```