Examining Factors Responsible for Heart Attacks

Objective:

Cardiovascular diseases are the leading cause of death globally. This analysis aims to identify the leading factors of Cardiovascular Diseases, using Logistic Regression model to predict the outcome of the test data.

Variable Descriptions:

age: age in years
sex: (1 = male; 0 = female)
cp: chest pain type

- Value 0: typical angina

  Value 1: atypical angina
  Value 2: non-anginal pain
  *Value 3: asymptomatic

trestbps: resting blood pressure (in mm Hg)
chol: serum cholestoral in mg/dl
fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
restecg: resting electrocardiographic results

- Value 0: normal
- Value 1: having ST-T wave abnormality (T wave inversions and/or-elevation or ST depression of > 0.05 mV)
- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

thalach: maximum heart rate achieved
exang: exercise induced angina (1 = yes; 0 = no)
oldpeak: ST depression induced by exercise relative to rest
slope: the slope of the peak exercise ST segment
ca: number of major vessels (0-3) colored by flourosopy
thal: thalassemia types:

- thal value 0 = Silent carrier
- thal value 1 = Mild carrier
- thal value 2 = Reverseable carrier
- thal value 3 = Fixed defect carrier

target: 0= less chance of heart attack, 1= more chance of heart attack

1. Import Modules & Data

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

df = pd.read_excel('heart data.xlsx')
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
```

- Dataset has 14 columns and 303 rows (inc header)
- There appears to be no missing values
- All columns contain int64 or float64 datatypes

In [2]:
```
df.shape
```

Out[2]:
```
(303, 14)
```

2. Data Wrangling

In [3]:
```
# check missing values
df.isnull().sum()
```

Out[3]:
```
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
```

In [4]:
```
# check for duplicates
df.duplicated().any()
```

Out[4]:
```
True
```

In [5]:
```
# drop duplicates and keep first occurance
df.drop_duplicates(keep='first', inplace=True)
df.reset_index(drop=True, inplace=True)
```

In [6]:
```
# view top 5 rows
df.head()
```

Out[6]:

	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

In [7]:
```
# check count of unique values
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
```

3. Exploratory Data Analysis

Central Tendencies & Data Distribution

In [8]:
```
# view statistics of data
df.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000

75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000

In [9]:
```python
# modes of df
modes = df.mode(axis=0, dropna=True)
modes
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	58.0	1.0	0.0	120.0	197	0.0	1.0	162.0	0.0	0.0	2.0	0.0	2.0	1.0
1	NaN	NaN	NaN	NaN	204	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	234	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [10]:
```python
# medians of df
medians = df.median()
medians
```

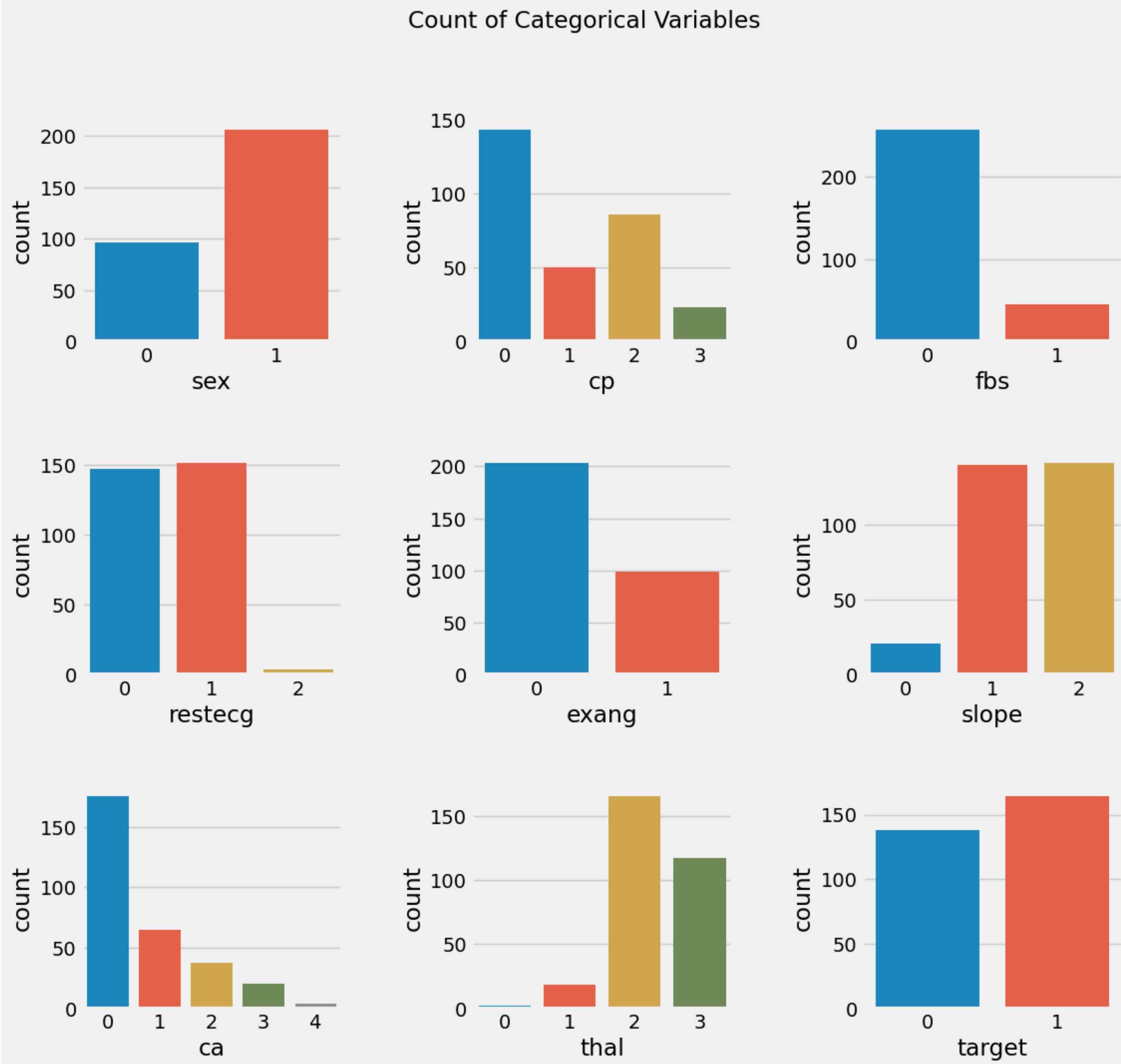
Out[10]:
```
age          55.5
sex           1.0
cp            1.0
trestbps    130.0
chol        240.5
fbs           0.0
restecg       1.0
thalach     152.5
exang         0.0
oldpeak       0.8
slope         1.0
ca            0.0
thal          2.0
target        1.0
dtype: float64
```

Countplots for Categorical Natured Variables

In [11]:
```python
plt.style.use('fivethirtyeight')
plt.figure(1, figsize=(12,11))
n =0

for x in ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']:
    n += 1
    plt.subplot(3,3,n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.countplot(data=df, x=x)

plt.suptitle('Count of Categorical Variables')
plt.show()
```



Observations:

- There are more Female patients as compared to Male patients
- Type 0 cp is most common
- fbs (blood fasting sugar) is much likely to be less than 120mg/dl
- Results 2 is very rare in restecg. Result 0 and 1 are most common
- There are more non-exercise induced anigmas
- There are more slope 1 and 2 occurences compared to slope 0
- The most common value of blood vessels is 0, and the least common blood vessel value is 4. This value 4 occurance may be an error, as it has nor been included into the data variable values description
- There are more value 1s of 'target', indicating there are more patients with likelihood of CVD in dataset.

Sanity Check:

In [12]:

```
# replacing value 4 'ca' to the nearest value 3, to adhere to variable despcription valu

df.ca = df['ca'].replace('4','3')
```

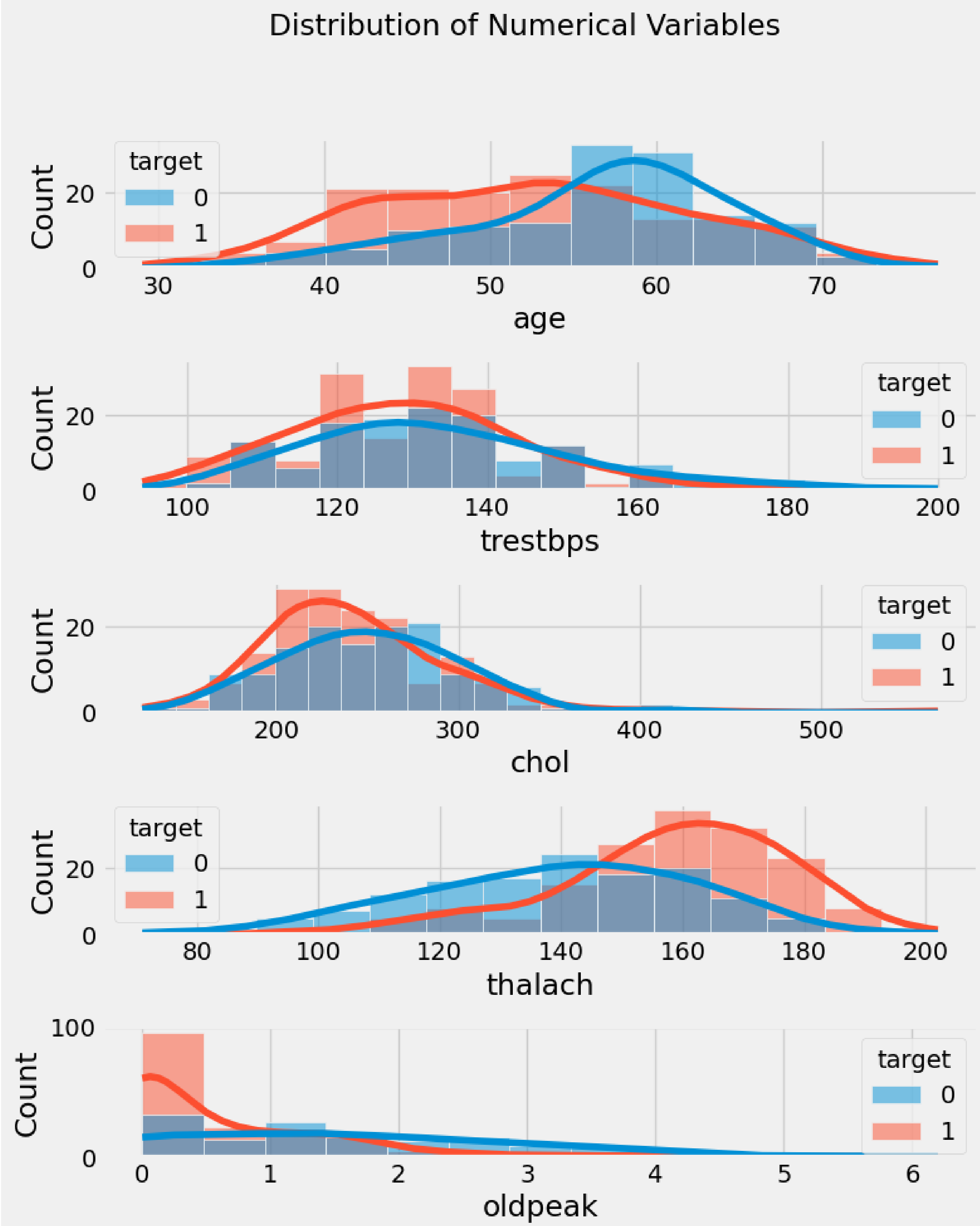
Distribution of Numerical Varaibles

In [13]:

```
plt.style.use('fast')
plt.figure(1, figsize=(8,10))
n =0

for x in ['age','trestbps','chol','thalach','oldpeak']:
    n += 1
    plt.subplot(5,1,n)
    plt.subplots_adjust(hspace=0.7, wspace=0.5)
    sns.histplot(data=df, x=x, kde=True, hue='target')

plt.suptitle('Distribution of Numerical Variables')
plt.show()
```

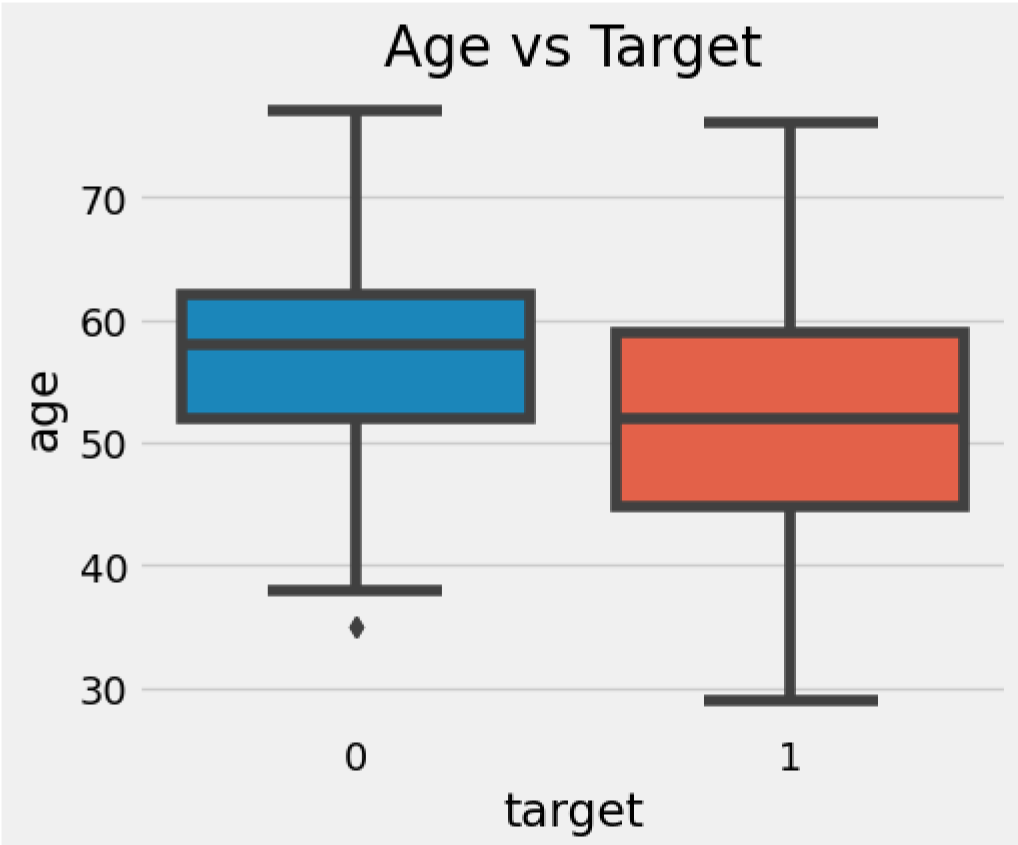


Observations:

- Patient age is concentrated around 45 and 65 range, peaking at late 50s
- Blood pressure is concentrated on the 120 -149 mark
- Chloesterol serum is denser in the 200 - 280 range
- The most common max heart rate achieved value is 160

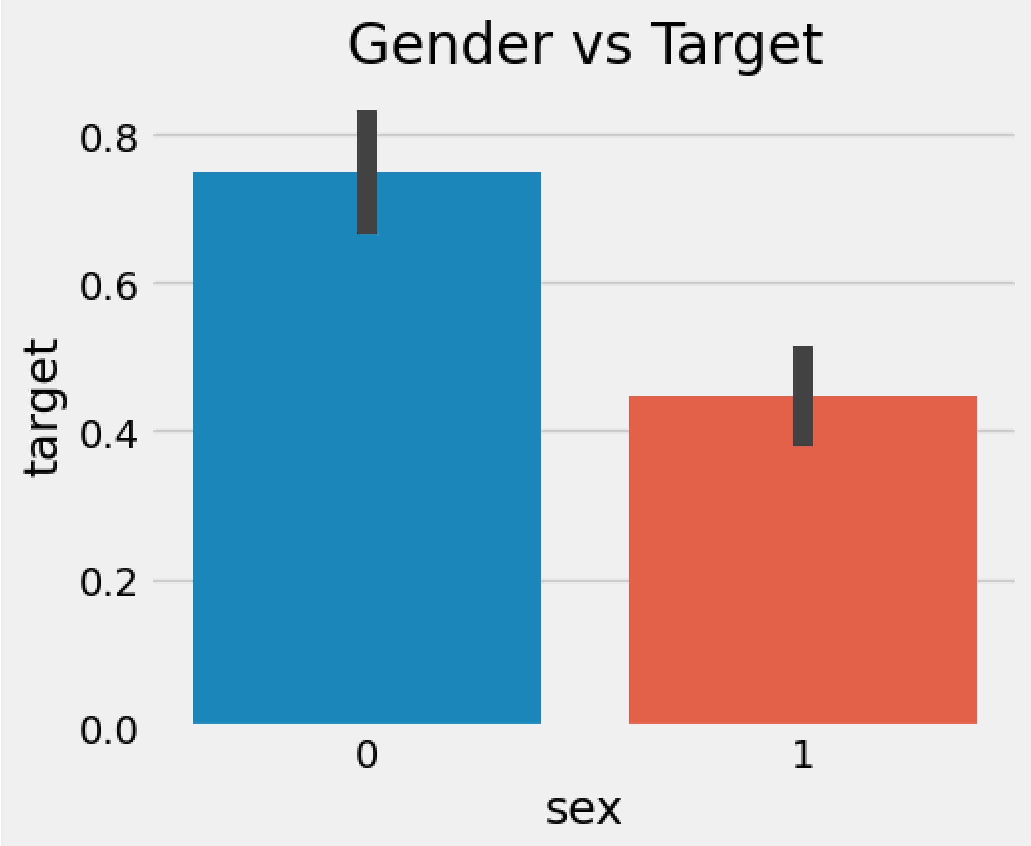
Bivariate Analysis

In [14]:
```
# age vs target
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5,4))
sns.boxplot(df, x='target', y='age')
plt.title('Age vs Target')
plt.show()
```



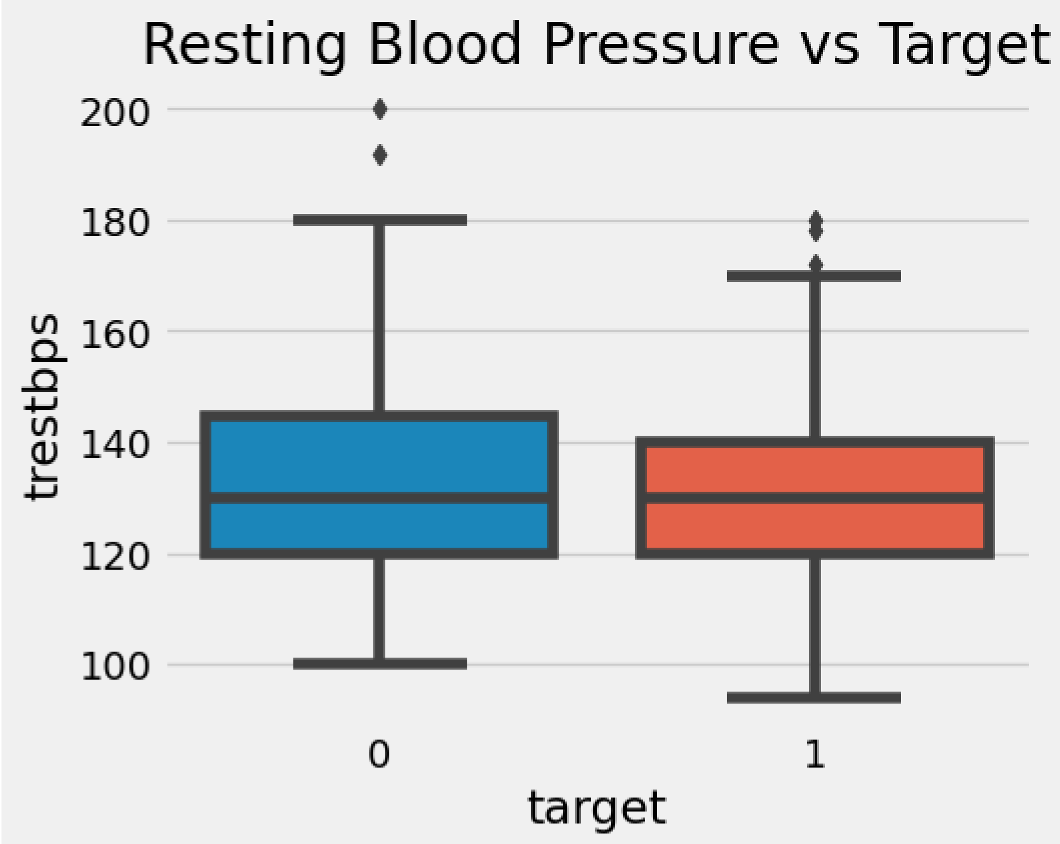
- **More chances of heart attacks occuring in the 45-60 age range**

In [15]:
```
# gender vs target
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5,4))
sns.barplot(df, x='sex', y='target')
plt.title('Gender vs Target')
plt.show()
```



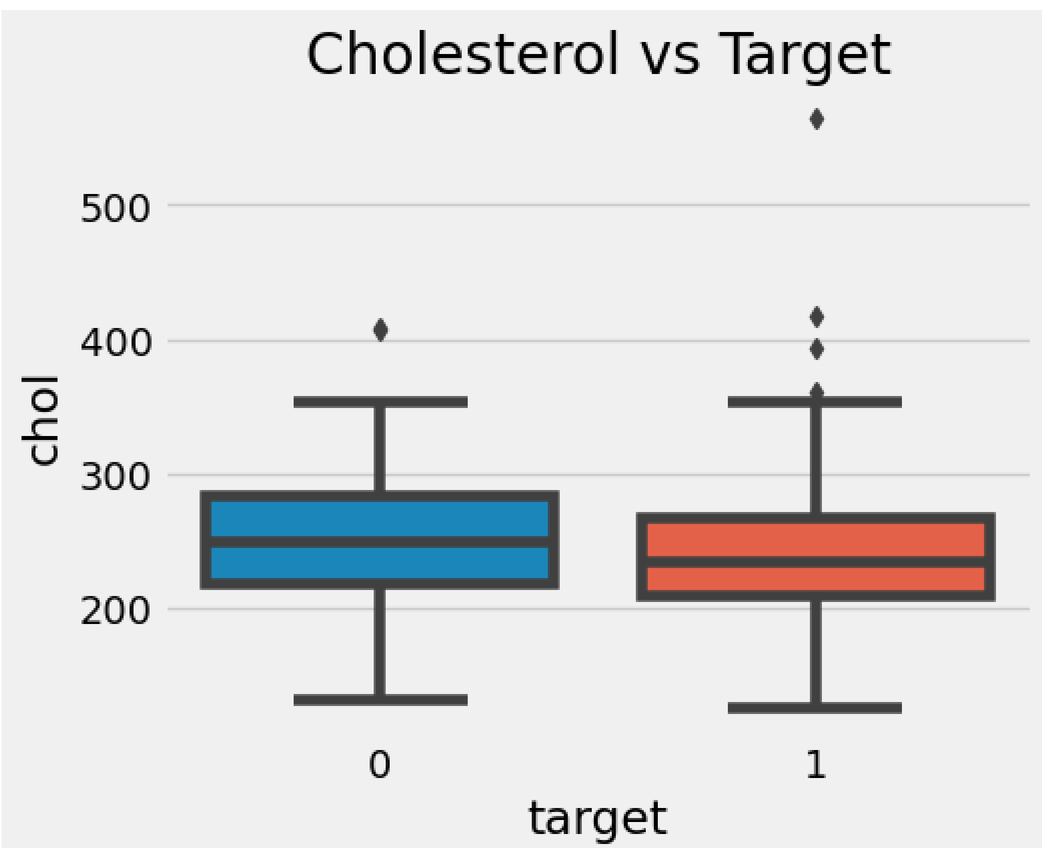
- **Male patients are more likely to get a heart attack**
- Female patients sit in the middle of target measurement

In [16]:
```
# trestbps vs target
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5,4))
sns.boxplot(df, x='target', y='trestbps')
plt.title('Resting Blood Pressure vs Target')
plt.show()
```



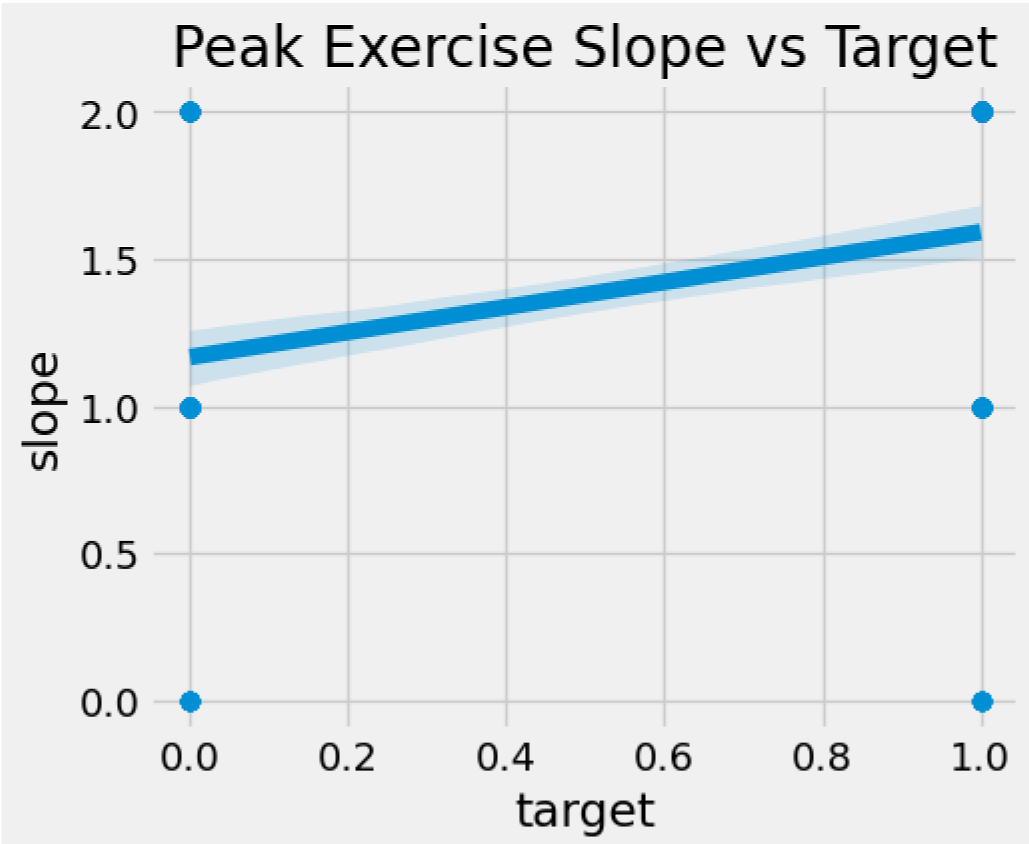
- More chances of a heart attack are seen in the resting blood pressure range of 120 and 140 mark

In [17]:
```
# chol vs target
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5,4))
sns.boxplot(df, x='target', y='chol')
plt.title('Cholesterol vs Target')
plt.show()
```



- Cholesterol values for targets appear almost the same.
- Outliers of more chance of heart attack is more prominent

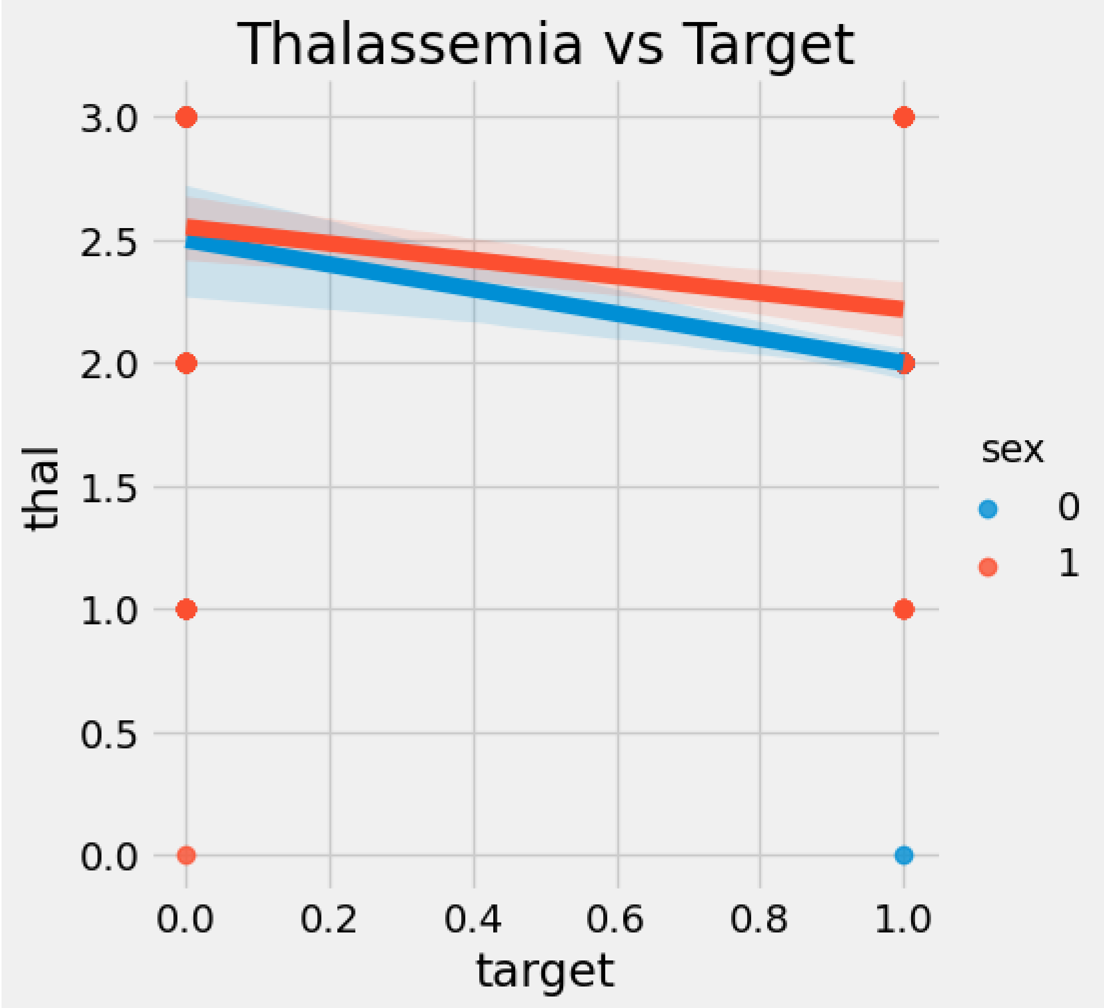
In [18]:
```
# slope vs target
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5,4))
sns.regplot(df, x='target', y='slope')
plt.title('Peak Exercise Slope vs Target')
plt.show()
```



- There is a positive trend between slope and cp.
- **This indicates the higher the slope of peak exercise, the more likely the CVD occurs**

In [19]:
```
# thal vs target
plt.style.use('fivethirtyeight')
plt.figure(figsize=(5,4))
sns.lmplot(df, x='target', y="thal", hue='sex')
plt.title('Thalassemia vs Target')
plt.show()
```

```
<Figure size 500x400 with 0 Axes>
```



- Thalassemia has a moderately negative relationship to target.
- The correlation between thal and target is -0.34
- **The higher value (in severity) of thal, the more likely the occurance of CVD.** This observation applies to both genders.

> Additonal variable description corrections need to be applied for 'thal' column as follows:
> thal value 0 = Silent carrier
> thal value 1 = Mild carrier
> thanl value 2 = Reverseable carrier
> thal value 3 = Fixed defect carrier

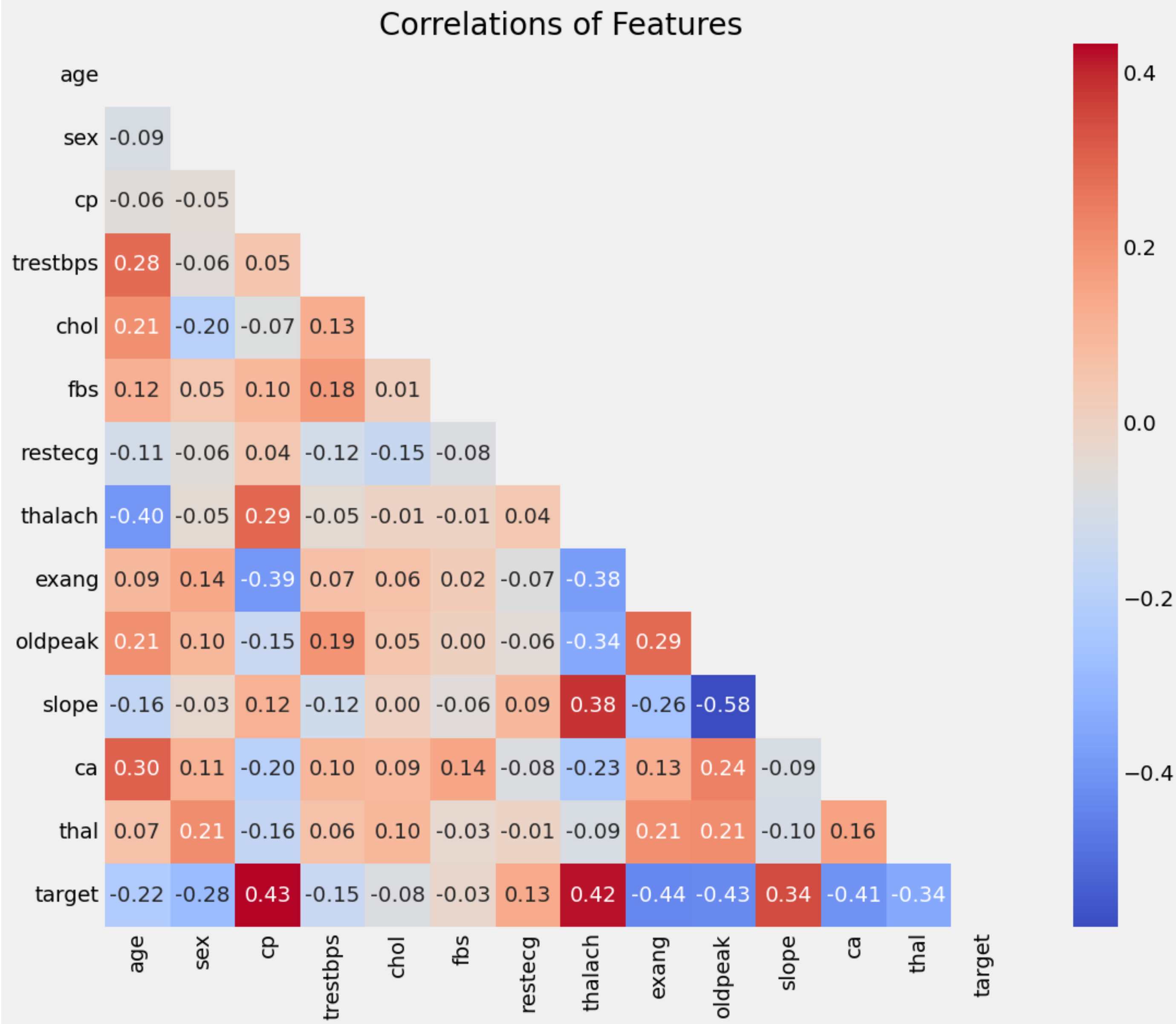
Correlations of Variables:

Understanding the relationships between features

In [20]:

```
# view correlations of features with heatmap

plt.figure(figsize=(12,10))
df_corr = df.corr()
mask = np.triu(np.ones_like(df_corr))
sns.heatmap(df_corr, cmap='coolwarm', annot=True, mask=mask, fmt='.2f')
plt.title('Correlations of Features')
plt.show()
```



Observations:

- **'target' is positively correlated to 'cp', 'thalach', 'slope', by 0.43, 0.42 and 0.34 respectively. This may indicate that the higher the value of 'cp', 'thalach' and 'slope', the more likely the occurence of an heart attack.**
- There is a negative correlation -0.58 between 'slope' and 'oldpeak'
- 'target' has negative correlations with ''exang', 'oldpeak', and 'ca' by -0.44, -0.43 and -0.41, respcetively.
- 'thalach' has negative correlations with 'exang' and 'oldpeak', of -0.38 and -0.34 respectively
- 'exang' and 'oldpeak' have a correlation of 0.29
- 'oldpeak' and 'ca' have acorrelation of 0.24

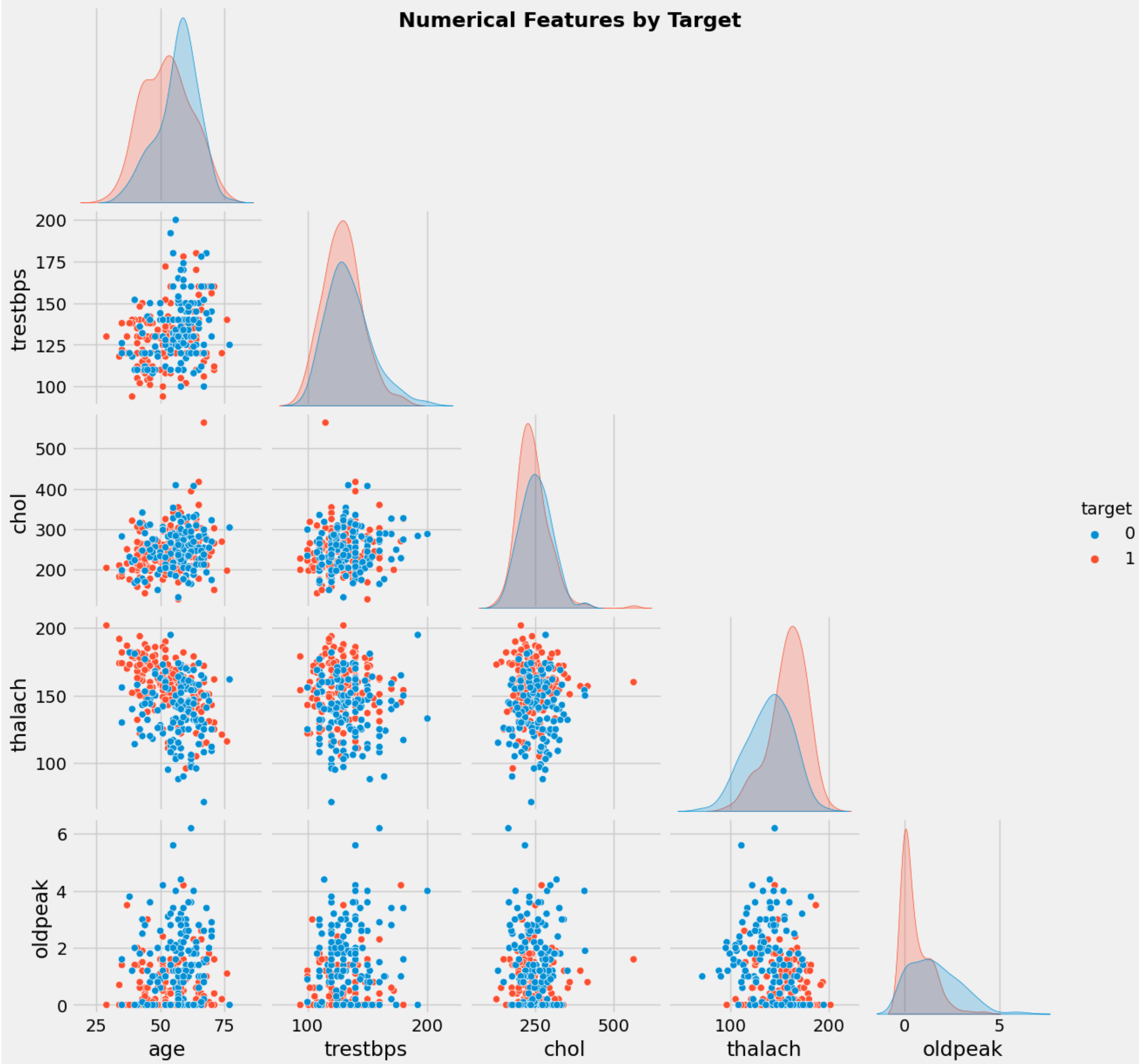
Pairplot of All Features

In [21]:
```
# pairplot of entire df
plt.style.use('fast')
sns.pairplot(df, hue='target', corner=True)
plt.suptitle('Pairplot of Heart Dataset', fontsize='35', fontweight='heavy')
plt.show()
```

Pairplot of Numerical Features by Target

In [22]:
```
num_data = ['age','trestbps','chol','thalach','oldpeak', 'target']
```

In [23]:
```
# Numerical Features
plt.style.use('fast')
sns.pairplot(df[num_data], hue='target', corner=True)
plt.suptitle('Numerical Features by Target', fontsize='17', fontweight='heavy')
plt.show()
```



In [24]:
```
df.to_excel('cleaneddata.xlsx')
```

4. Model Development: Logistic Regression

i. Import Modelling Modules

In [25]:
```
# import modelling modules
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score
```

ii. Select Features for Analysis

In [26]:
```
# declare feature selection:
# trestbps, chol, fbs and restecg will be dropped due to their very low correlation valu

x = df.drop(columns=['trestbps', 'chol', 'fbs', 'restecg', 'target'])
y = df['target']
```

iii. Assess Indicator Variables

In [27]:
```
# confirm features are already encoded as per the variable description
print(df.apply(lambda col: col.unique()))
```

```
age         [63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 5...
sex                                                     [1, 0]
cp                                                [3, 2, 1, 0]
trestbps    [145, 130, 120, 140, 172, 150, 110, 135, 160, ...
chol        [233, 250, 204, 236, 354, 192, 294, 263, 199, ...
fbs                                                     [1, 0]
restecg                                              [0, 1, 2]
thalach     [150, 187, 172, 178, 163, 148, 153, 173, 162, ...
exang                                                   [0, 1]
oldpeak     [2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, ...
slope                                                [0, 2, 1]
ca                                             [0, 2, 1, 3, 4]
thal                                              [1, 2, 3, 0]
target                                                  [1, 0]
dtype: object
```

iv. Split data into Training & Test Sets

In [28]:
```
# split the dataset into training and test set

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42
print('Shape of Training & Testing Datasets:')
print("Train_x :",x_train.shape)
print("Test_x :",x_test.shape)
print("Train_y :",y_train.shape)
print("Test_y :",y_test.shape)
```

```
Shape of Training & Testing Datasets:
Train_x : (211, 9)
Test_x : (91, 9)
Train_y : (211,)
Test_y : (91,)
```

v. Feature Scaling with StandardScaler

In [29]:
```
# scale independant features only
# dependent variable is already valued at 0-1
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

vi. Fit Logistic Regression to Training Set

In [30]:
```
# fit trainining set into logreg object
logreg = LogisticRegression(random_state=0)
logreg.fit(x_train, y_train)
```

Out[30]:
```
LogisticRegression(random_state=0)
```

vii. Predict with Test set

In [31]:
```
predict = logreg.predict(x_test)
```

```
# predicted values
predict
```

Out[31]:
```
array([0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
       1, 0, 1], dtype=int64)
```

viii. Check Accuracy

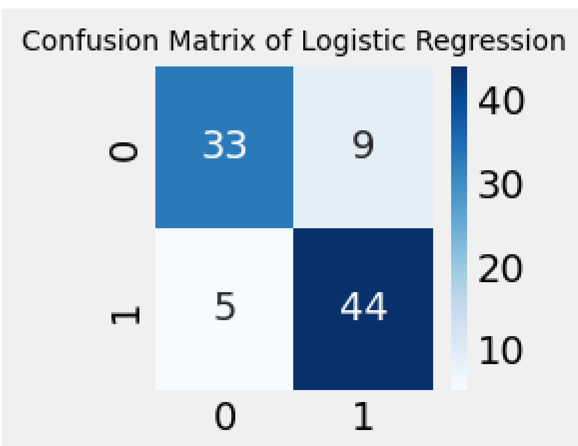
In [32]:
```
print('Logistic Regression Accuracy Score is:', accuracy_score(y_test, predict)*100, '%'
```

```
Logistic Regression Accuracy Score is: 84.61538461538461 %
```

ix. Fit Confusion Matrix into Test Set

In [33]:
```
# fit confusion matrix into test set
conf_matrix = confusion_matrix(y_test, predict)
```

In [34]:
```
# visualise confusion matrix
plt.figure(figsize=(2,2))
sns.heatmap(conf_matrix, annot=True, cmap='Blues')
plt.title('Confusion Matrix of Logistic Regression', fontsize='10')
plt.show()
```



x. Model Performance

In [35]:
```
# Calculating False Positives (FP), False Negatives (FN), True Positives (TP) & True Neg

def model_performance(conf_matrix):

    FP = conf_matrix.sum(axis=0) - np.diag(conf_matrix)
    FN = conf_matrix.sum(axis=1) - np.diag(conf_matrix)
    TP = np.diag(conf_matrix)
    TN = conf_matrix.sum() - (FP + FN + TP)

    # Recall or true positive rate
    TPR = TP/(TP+FN)
    print ("The Recall (True Positive rate) per class is: ",TPR)


    # Precision or positive predictive value
    PPV = TP/(TP+FP)
    print ("The Precision per class is: ",PPV)

    # Overall accuracy
```

```
    ACC = (TP+TN)/(TP+FP+FN+TN)
    print ("The Accuracy of each class is", ACC)
    print("")

    ##Total averages :
    print ("The average Recall is: ",TPR.sum()/2)
    print ("The average Precision is: ",PPV.sum()/2)
    print ("The average Accuracy is", ACC.sum()/2)
```

Model Performance: Recall, Precision and Accuracy

In [36]:
```
model_performance(conf_matrix)
```

```
The Recall (True Positive rate) per class is:  [0.78571429 0.89795918]
The Precision per class is:  [0.86842105 0.83018868]
The Accuracy of each class is [0.84615385 0.84615385]

The average Recall is:  0.8418367346938775
The average Precision is:  0.849304865938431
The average Accuracy is 0.8461538461538461
```

5. Tableau Dashboard

Tableau Dashboard Link:
https://public.tableau.com/views/HeartAttackFactors_16738268795260/Dashboard1?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link