03_heart

October 22, 2021

1 Heart Failure Prediction Dataset

Activity performed for the discipline "Data Mining"

Matheus Pimenta

1.0.1 Accessed in Oct. 2021

1.0.2 Available here

1.0.3 Kaggle

1.1 Description

1.1.1 Context

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Four out of 5CVD deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age. Heart failure is a common event caused by CVDs and this dataset contains 11 features that can be used to predict a possible heart disease.

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

1.1.2 Source

This dataset was created by combining different datasets already available independently but not combined before. In this dataset, 5 heart datasets are combined over 11 common features which makes it the largest heart disease dataset available so far for research purposes. The five datasets used for its curation are:

- Cleveland: 303 observations
- Hungarian: 294 observations
- Switzerland: 123 observations
- Long Beach VA: 200 observations
- Stalog (Heart) Data Set: 270 observations Total: 1190 observations Duplicated: 272 observations Final dataset: 918 observations

Every dataset used can be found under the Index of heart disease datasets from UCI Machine Learning Repository

1.1.3 For more information visit the Kaggle site.

1.2 Importing the libraries

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy.stats import pearsonr
import seaborn as sns
```

1.3 Load the dataset using Pandas API

[ ]:
```
df = pd.read_csv('heart_3.csv',
                 header = 0)
```

1.4 Getting some information about the dataset

[ ]:
```
print("Dataset shape:{}\nThere are {} rows and {} columns.".format(
    df.shape, df.shape[0], df.shape[1]))
df.head(3)
```

```
Dataset shape:(1025, 14)
There are 1025 rows and 14 columns.
```

[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0

There are 1025 rows and 14 columns, that is, 1025 samples and 13 features + 1 output label, in this case, the column "HeartDisease" represents if the sample has heart diseases.

[ ]:
```
df.dtypes
```

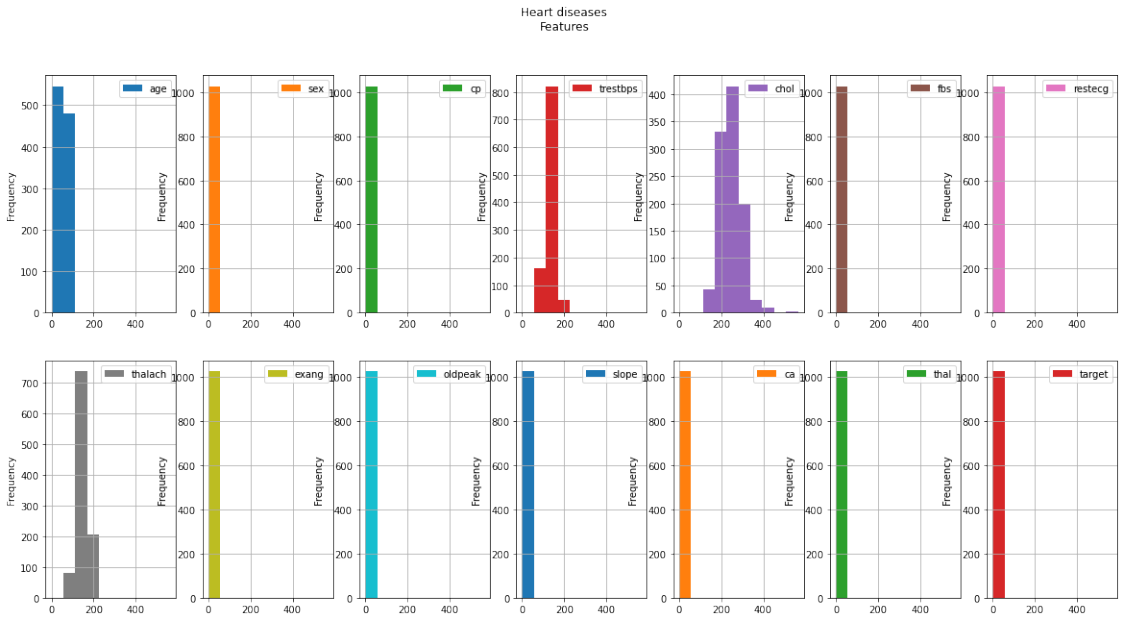
[ ]:
```
age          int64
sex          int64
cp           int64
trestbps     int64
chol         int64
fbs          int64
restecg      int64
thalach      int64
```

```
exang          int64
oldpeak      float64
slope          int64
ca             int64
thal           int64
target         int64
dtype: object
```

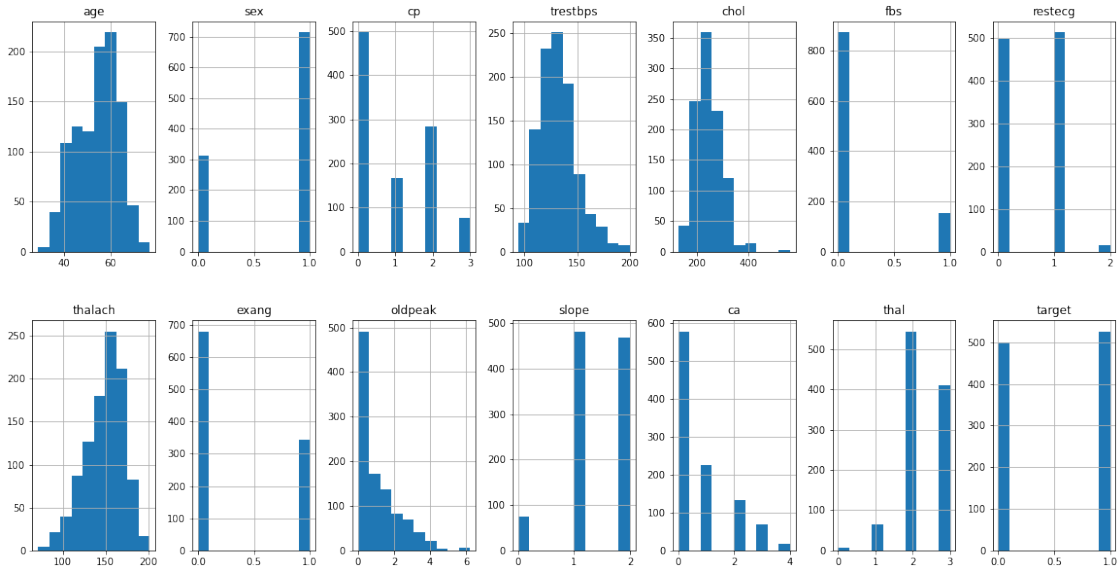
1.5 Histograms

Note: **There's an error on the 'plot' function. Even that 'sharex' and 'sharey' options are disabled, the plot doesn't apply the changes.**

[ ]:
```
ax = df.plot(kind='hist',
             sharex=False,
             sharey=False,
             subplots=True,
             figsize = (20,10),
             layout=(2, 7),
             grid=True,
             title='Heart diseases\nFeatures')
```

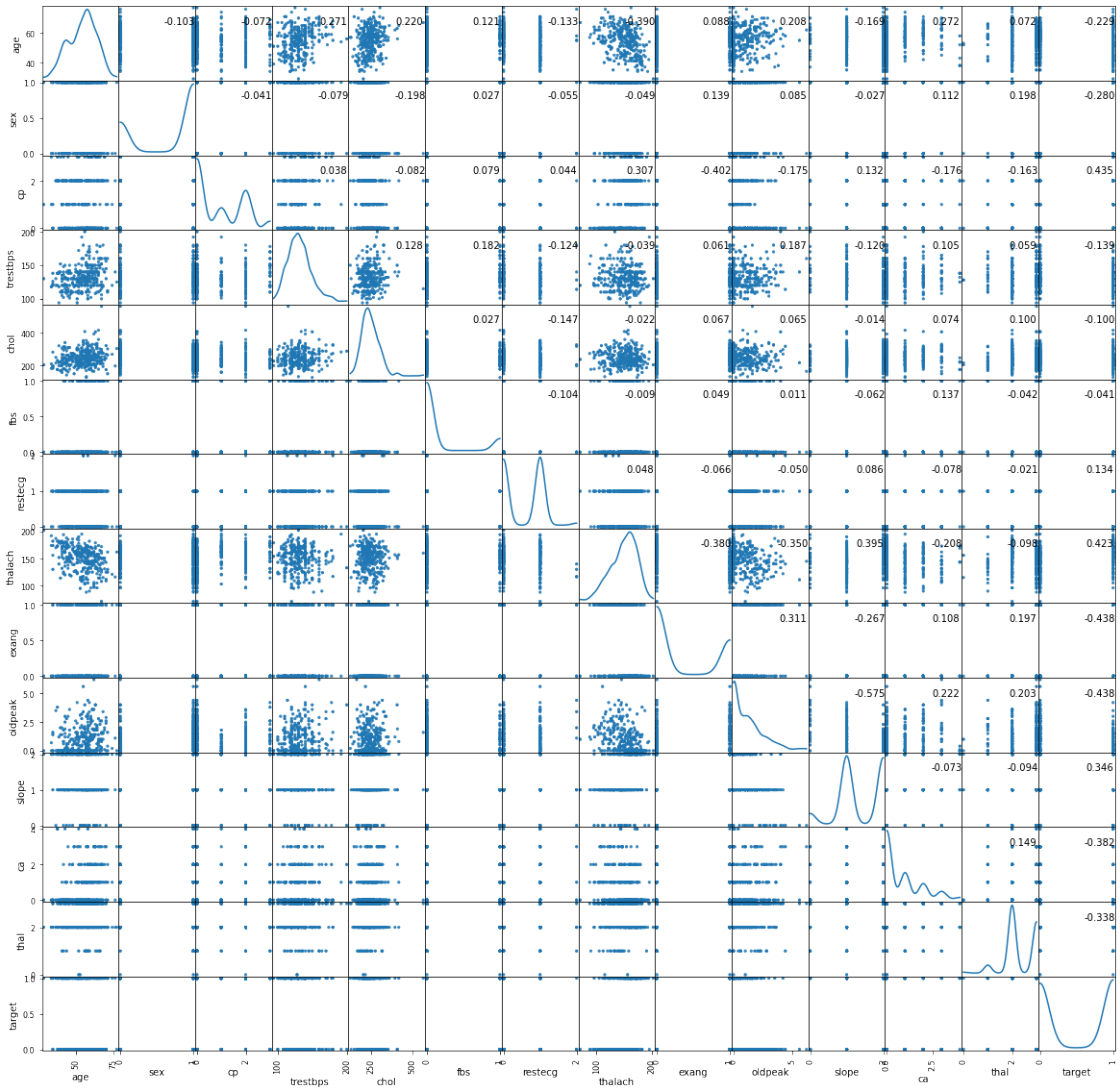


[ ]:
```
ax = df.hist(figsize = (20,10),
             layout=(2, 7),
             grid=True)
```

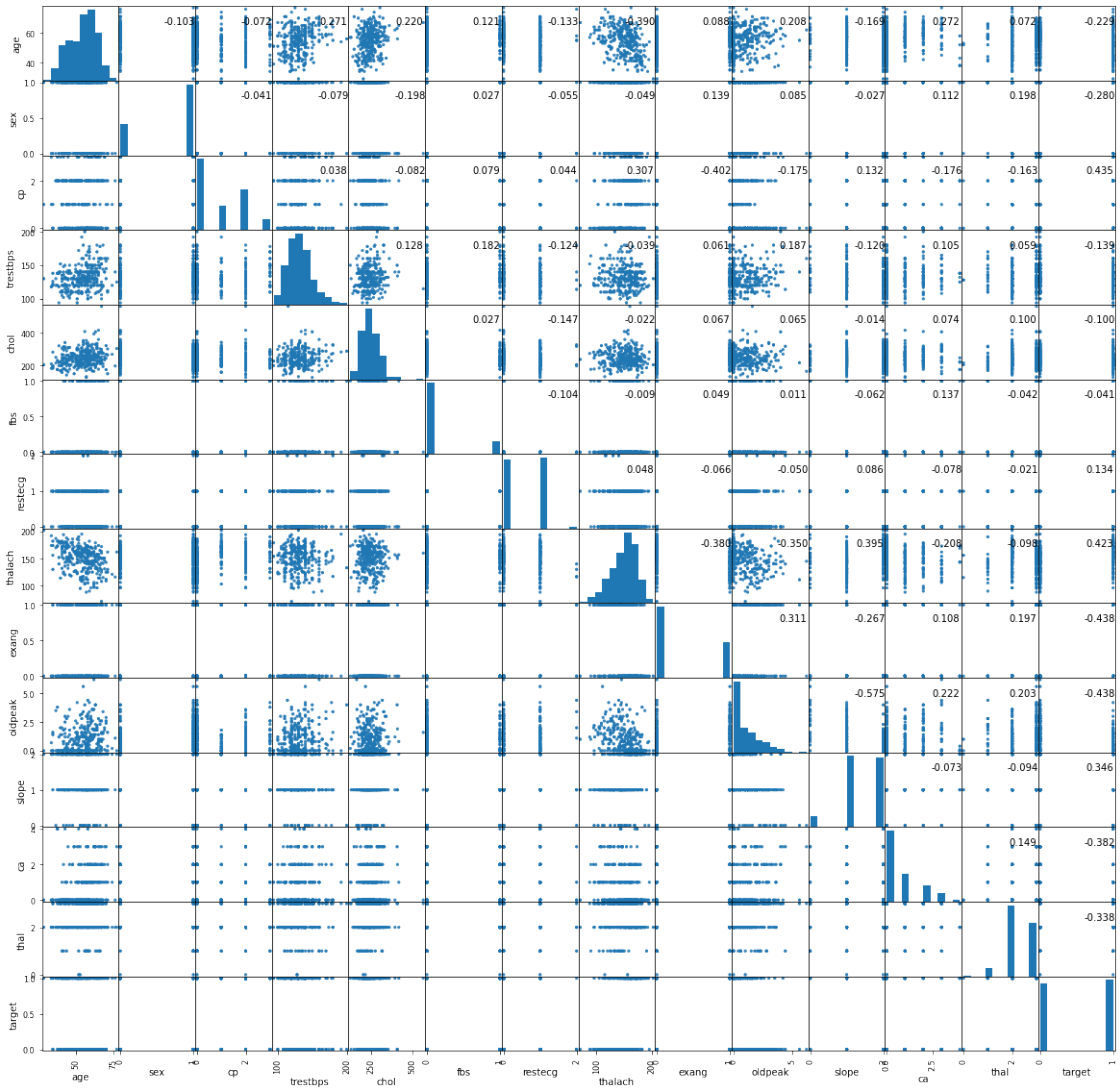


1.6 Scatter-matrix

[ ]:
```
ax = pd.plotting.scatter_matrix(df,
                                diagonal = "kde",
                                figsize=(20,20))
corr = np.asmatrix(df.corr())
for i, j in zip(*plt.np.triu_indices_from(ax, k=1)):
    ax[i, j].annotate("%.3f" %corr[i,j], (0.8, 0.8), xycoords='axes fraction',␣
 ↪ha='center', va='center')
plt.show()
```



[ ]:
```
ax = pd.plotting.scatter_matrix(df,
                                diagonal = "hist",
                                figsize=(20,20))
corr = np.asmatrix(df.corr())
for i, j in zip(*plt.np.triu_indices_from(ax, k=1)):
    ax[i, j].annotate("%.3f" %corr[i,j], (0.8, 0.8), xycoords='axes fraction',␣
 ↪ha='center', va='center')
plt.show()
```



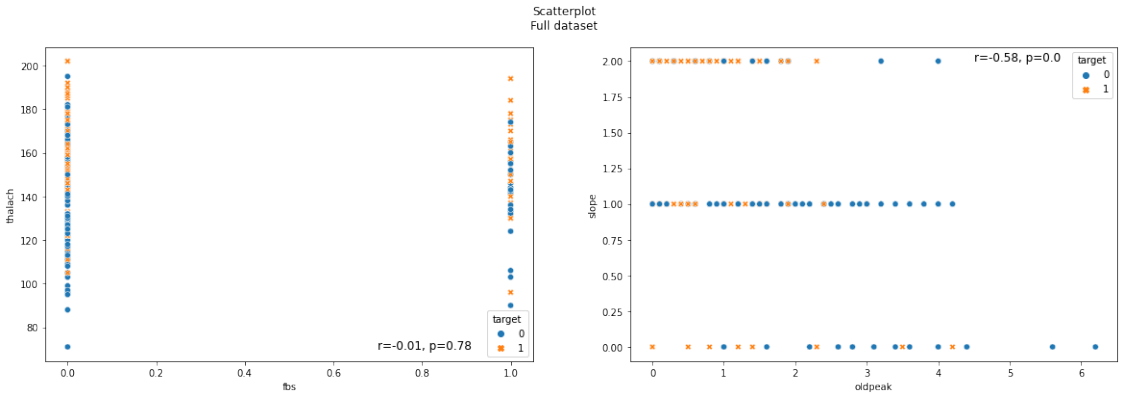
[ ]:
```
g = sns.pairplot(df,
                 hue='target')
g._legend.set_title("Scatter-matrix")
```

1.7 Scatterplot of two features.

```
[ ]: corr_1 = pearsonr(df['fbs'], df['thalach'])
corr_1 = [np.round(c, 2) for c in corr_1]
text_1 = 'r=%s, p=%s' % (corr_1[0], corr_1[1])
corr_2 = pearsonr(df['oldpeak'], df['slope'])
corr_2 = [np.round(c, 2) for c in corr_2]
text_2 = 'r=%s, p=%s' % (corr_2[0], corr_2[1])
fig, ax = plt.subplots(1,2, figsize = (20,6))
fig.suptitle("Scatterplot\nFull dataset")
sns.scatterplot(data=df,
                    x="fbs",
                    y="thalach",
```

```
                      hue="target",
                      ax = ax[0],
                      style="target")
ax[0].text(0.7, 70, text_1, fontsize=12)
sns.scatterplot(data=df,
                x="oldpeak",
                y="slope",
                hue="target",
                ax = ax[1],
                style="target")
ax[1].text(4.5, 2, text_2, fontsize=12)
```

[ ]:
```
Text(4.5, 2, 'r=-0.58, p=0.0')
```



1.8 Splitting the dataset into two sets

[ ]:
```
df_simple_sample = df.sample(frac=0.3) # simple sample - 30%
X_train, X_test, y_train, y_test = train_test_split(df.drop('target',axis=1),
                                                    df['target'],
                                                    stratify=df['target'],
                                                    test_size=0.30) #␣
 ↪stratified sample - 30%
X_test = X_test.join(y_test)
X_train = X_train.join(y_train)
```

1.9 Simple sample - describe

[ ]:
```
df_simple_sample.describe().T
```

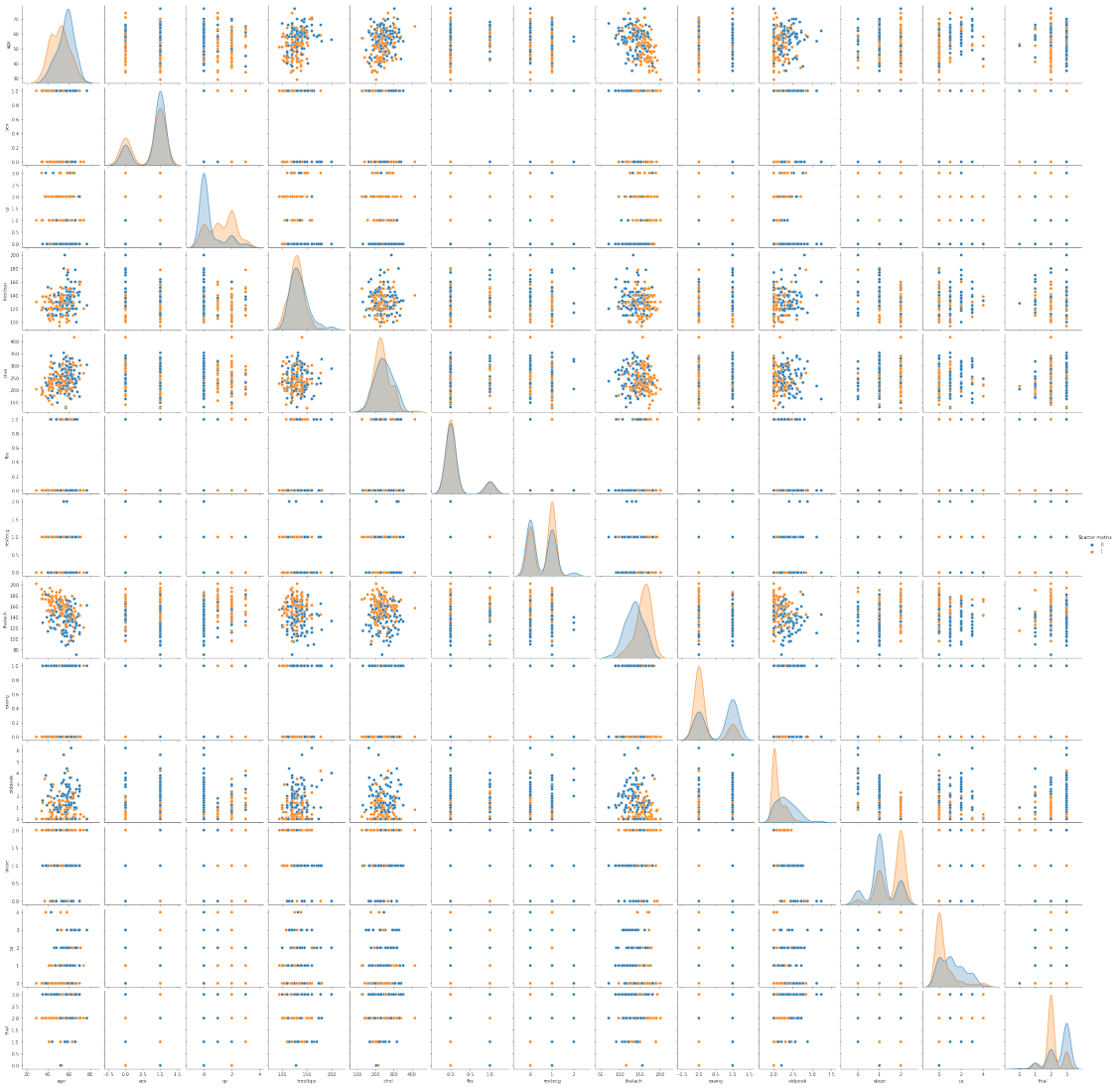
[ ]:

	count	mean	std	min	25%	50%	75%	max
age	308.0	53.844156	9.026656	29.0	47.00	54.0	61.00	77.0
sex	308.0	0.730519	0.444412	0.0	0.00	1.0	1.00	1.0

cp	308.0	0.876623	0.997247	0.0	0.00	0.0	2.00	3.0
trestbps	308.0	131.003247	17.075898	94.0	120.00	130.0	140.00	200.0
chol	308.0	241.029221	46.473093	126.0	209.50	234.5	269.25	417.0
fbs	308.0	0.146104	0.353785	0.0	0.00	0.0	0.00	1.0
restecg	308.0	0.551948	0.529793	0.0	0.00	1.0	1.00	2.0
thalach	308.0	147.003247	24.326565	71.0	131.75	149.5	166.25	202.0
exang	308.0	0.383117	0.486938	0.0	0.00	0.0	1.00	1.0
oldpeak	308.0	1.115584	1.250909	0.0	0.00	0.8	1.90	6.2
slope	308.0	1.350649	0.635871	0.0	1.00	1.0	2.00	2.0
ca	308.0	0.847403	1.100530	0.0	0.00	0.0	1.00	4.0
thal	308.0	2.314935	0.636694	0.0	2.00	2.0	3.00	3.0
target	308.0	0.503247	0.500803	0.0	0.00	1.0	1.00	1.0

1.10 Simple sample - Scatterplots

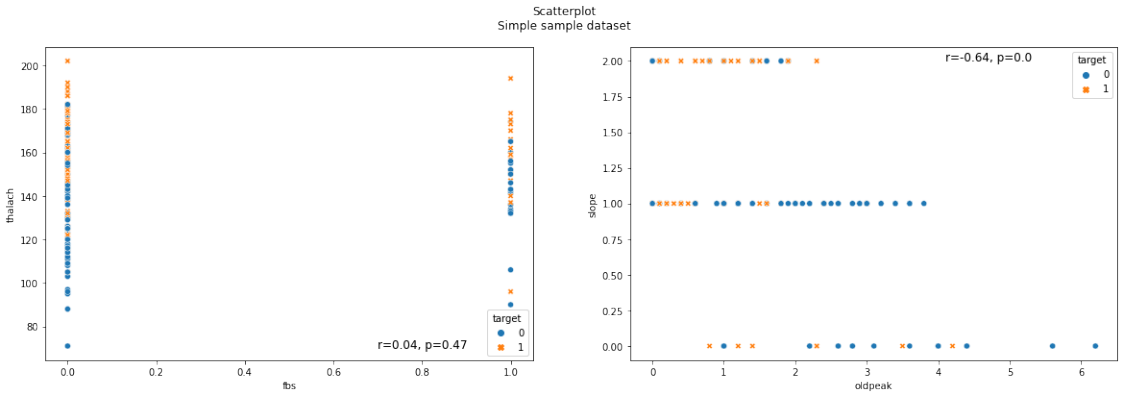
[ ]:
```
g = sns.pairplot(df_simple_sample,
                 hue='target')
g._legend.set_title("Scatter-matrix")
```



[ ]:
```
corr_1 = pearsonr(df_simple_sample['fbs'], df_simple_sample['thalach'])
corr_1 = [np.round(c, 2) for c in corr_1]
text_1 = 'r=%s, p=%s' % (corr_1[0], corr_1[1])
corr_2 = pearsonr(df_simple_sample['oldpeak'], df_simple_sample['slope'])
corr_2 = [np.round(c, 2) for c in corr_2]
text_2 = 'r=%s, p=%s' % (corr_2[0], corr_2[1])
fig, ax = plt.subplots(1,2, figsize = (20,6))
fig.suptitle("Scatterplot\nSimple sample dataset")
sns.scatterplot(data=df_simple_sample,
                        x="fbs",
                        y="thalach",
                        hue="target",
                        ax = ax[0],
                        style="target")
```

```
ax[0].text(0.7, 70, text_1, fontsize=12)
sns.scatterplot(data=df_simple_sample,
                x="oldpeak",
                y="slope",
                hue="target",
                ax = ax[1],
                style="target")
ax[1].text(4.1, 2, text_2, fontsize=12)
```

[ ]: Text(4.1, 2, 'r=-0.64, p=0.0')



1.11 Stratified sample - describe

[ ]:
```
X_test.describe().T
```

[ ]:

	count	mean	std	min	25%	50%	75%	max
age	308.0	54.610390	8.943406	29.0	48.00	55.0	61.0	77.0
sex	308.0	0.727273	0.446087	0.0	0.00	1.0	1.0	1.0
cp	308.0	0.902597	1.009850	0.0	0.00	1.0	2.0	3.0
trestbps	308.0	130.866883	17.071944	94.0	120.00	130.0	140.0	200.0
chol	308.0	246.935065	54.126908	126.0	211.75	239.5	278.0	564.0
fbs	308.0	0.136364	0.343733	0.0	0.00	0.0	0.0	1.0
restecg	308.0	0.519481	0.519594	0.0	0.00	1.0	1.0	2.0
thalach	308.0	148.954545	23.028333	71.0	132.00	152.0	166.0	202.0
exang	308.0	0.337662	0.473682	0.0	0.00	0.0	1.0	1.0
oldpeak	308.0	0.993506	1.135462	0.0	0.00	0.6	1.6	5.6
slope	308.0	1.376623	0.615520	0.0	1.00	1.0	2.0	2.0
ca	308.0	0.808442	1.036308	0.0	0.00	0.0	1.0	4.0
thal	308.0	2.331169	0.609963	0.0	2.00	2.0	3.0	3.0
target	308.0	0.512987	0.500645	0.0	0.00	1.0	1.0	1.0

1.12 Stratified sample - Scatterplots

[ ]:
```
g = sns.pairplot(X_test,
                 hue='target')
g._legend.set_title("Scatter-matrix")
```

[ ]:
```
corr_1 = pearsonr(X_test['fbs'], X_test['thalach'])
corr_1 = [np.round(c, 2) for c in corr_1]
text_1 = 'r=%s, p=%s' % (corr_1[0], corr_1[1])
corr_2 = pearsonr(X_test['oldpeak'], X_test['slope'])
corr_2 = [np.round(c, 2) for c in corr_2]
text_2 = 'r=%s, p=%s' % (corr_2[0], corr_2[1])
fig, ax = plt.subplots(1,2, figsize = (20,6))
fig.suptitle("Scatterplot\nStratified dataset")
```

```
sns.scatterplot(data=X_test,
                    x="fbs",
                    y="thalach",
                    hue="target",
                    ax = ax[0],
                    style="target")
ax[0].text(0.7, 70, text_1, fontsize=12)
sns.scatterplot(data=X_test,
                x="oldpeak",
                y="slope",
                hue="target",
                ax = ax[1],
                style="target")
ax[1].text(4, 2, text_2, fontsize=12)
```

[ ]: Text(4, 2, 'r=-0.55, p=0.0')