PREDICTING HEART DISEASE USING UCI DATASET

STEP1: Exploratory Data Analysis

Import necessary libraries

In [3]:
```
import pandas as pd
import numpy as np #not using this yet
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import pearsonr
```

Load CSV Data

Columns:

- **age:** age in years
- **sex:**(1 = male; 0 = female)
- **cp:** chest pain type
- **trestbps:** resting blood pressure (in mm Hg on admission to the hospital)
- **chol:** serum cholestoral in mg/dl
- **fbs:** fasting blood sugar>120mg/dl, (1 = true; 0 = false)
- **restecg:** resting electrocardiographic results
- **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:** 3=normal, 6=fixed defect, 7=reversable defect
- **target:** 1 or 0 (predicted attribute)

In [4]:
```
data = pd.read_csv('/Users/ozlemkorucuoglu/Desktop/CODERGIRL/KaagleProjec
data.head()
```

Out[4]:

	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

As can be seen above, our data is composed of categorical and continuous data. Our categoricl variables are : sex, cp, fbs,restecg, exang,slope, ca, thal.

Data Visualisation

Summary Statistics. Percentiles can help identify the range for most of the data Averages and medians can describe central tendency Correlations can indicate strong relationships

In [228]:
```
data.describe()
```

Out[228]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202

In [229]:
```
Data_cont = data[['age', 'trestbps', 'chol', 'thalach', 'oldpeak']].copy(
Data_cat = data[['sex', 'cp', 'fbs', 'restecg', 'exang','slope', 'ca', 't
```

Correlation Between Continuous Variables in our data. Plotting pairwise correlations across our variables can show us if some of our variables explain the same variance.

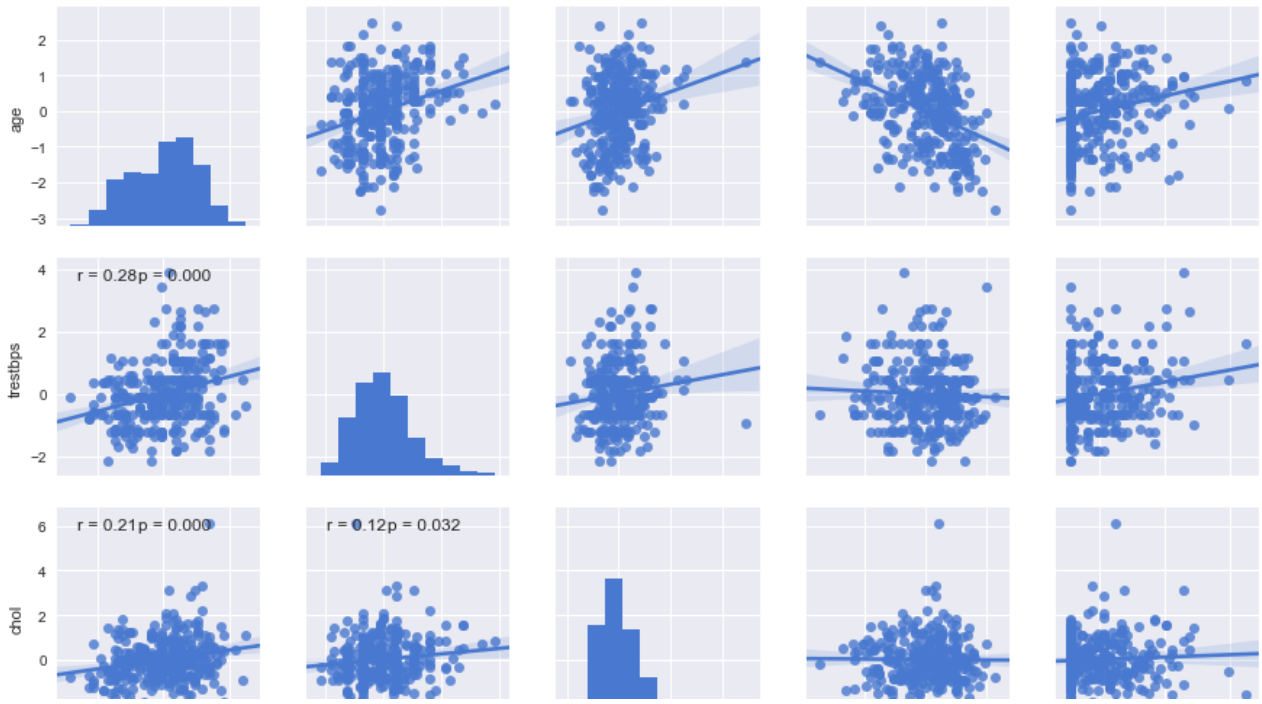
- First apply z-transformation to the data (optional, see below * for explanation)
- Across our variables, the maximum correlation is .4 (in negative direction), representing a weak correlation and assures that each of these variables individually carry plenty of unique variance (unique information). If there were two variables with close to perfect correlation (i.e. r=.9), we may consider to reduce dimensionality of these variables by applying PCA, to deal with a possible problem of overfitting.
-Together with the pairplot, the distribution of our variables are also plotted. In the following section, we wil further investigate these distributions in detail to ckeck for outliers.

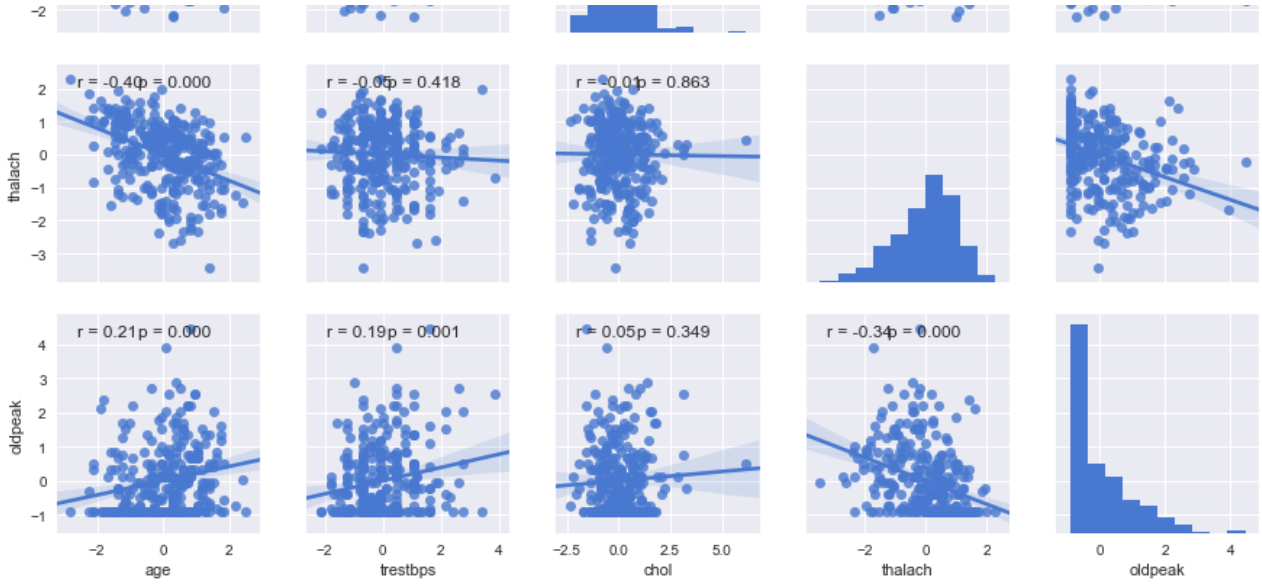
*Note that z-transformation does not change our pairwise scatterplots or the magnitude of our correlations (neither r-values or p-values), only the scale of x-axis and y-axis across variables are more comparible.

In [230]:
```
Data_cont_z = (Data_cont - Data_cont.mean()) / Data_cont.std()

def corrfunc(x, y, **kws):
    (r, p) = pearsonr(x, y)
    ax = plt.gca()
    ax.annotate("r = {:.2f} ".format(r),
                xy=(.1, .9), xycoords=ax.transAxes)
    ax.annotate("p = {:.3f}".format(p),
                xy=(.4, .9), xycoords=ax.transAxes)

graph = sns.pairplot(Data_cont_z, kind="reg")
graph.map_lower(corrfunc)
plt.show()
```



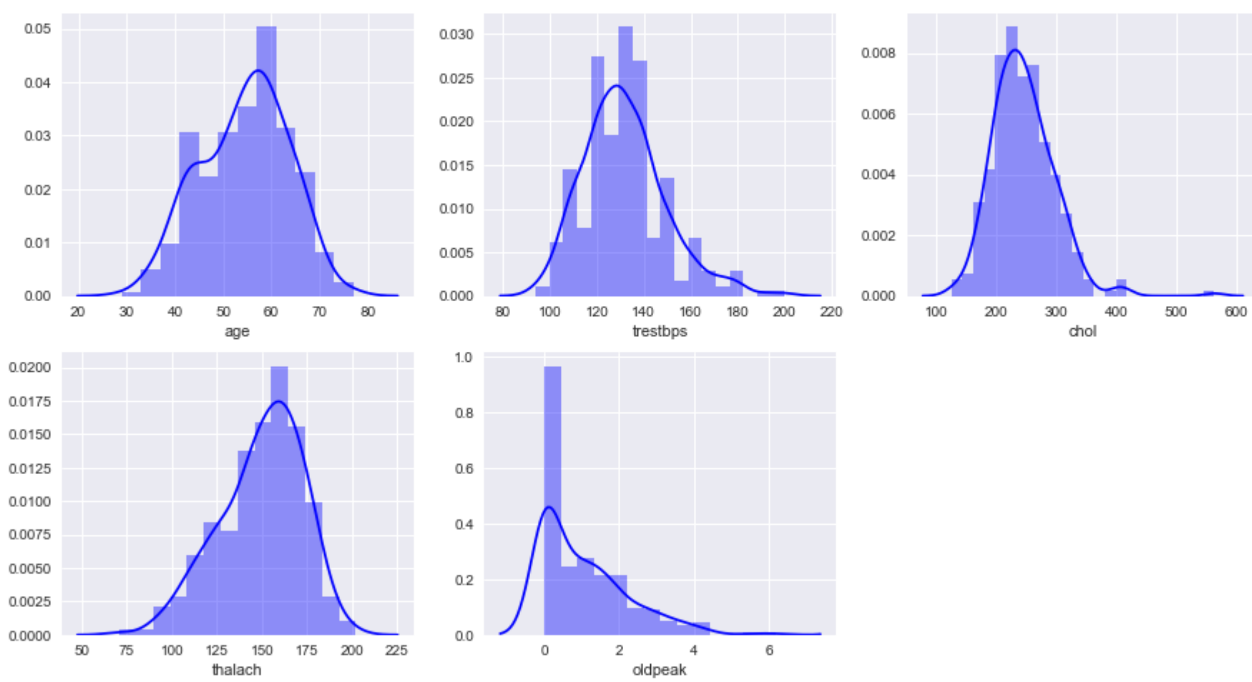


Check distribution of our continuous data. The above pairplot already prints the scatterplots of our data but here we take a closer look at the distribution, check for kurtosis, skewness and binomiality in data.

In [231]:
```
fig, axs = plt.subplots(nrows = 3, ncols=2, figsize = (15,8))

plot_number = 1
for i, x in enumerate(Data_cont):
    ax = plt.subplot(2, 3, plot_number)
    sns.distplot(Data_cont[x], color='blue', ax=ax)
    plot_number=plot_number+1
plt.show()
```

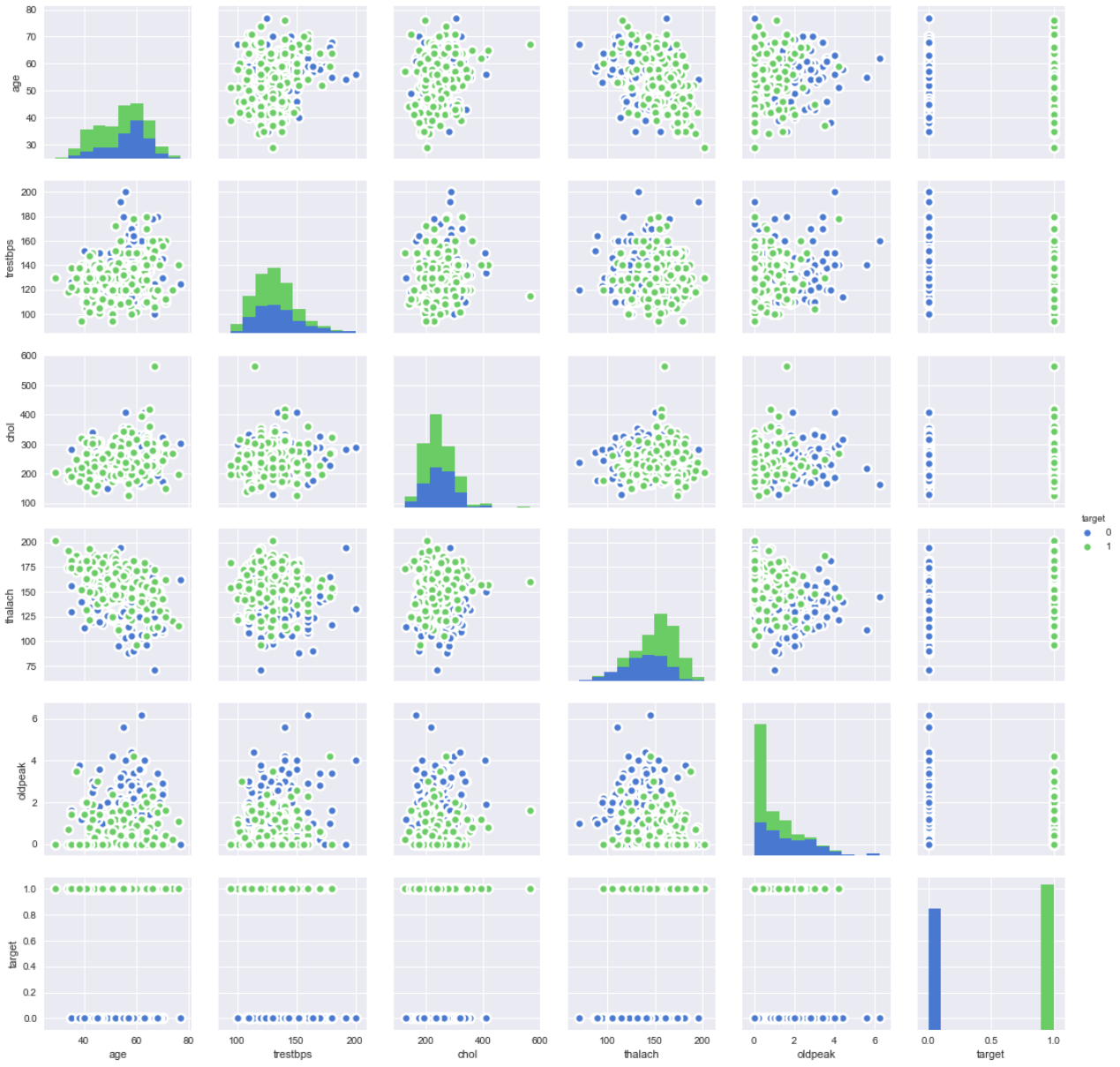
```
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced b
y the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced b
y the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced b
y the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced b
y the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced b
y the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



Although the first 4 of the variables seem to be slightly skewed, given our sample size, that level of skewness is not too concerning at this point. Moreover not all models require data with normal distribution, for example skewness affects the regression intercept and coefficients associated with the model. Therefore if we need any transformation of our variables depends on which model we decide to apply on our data.
The only variable with concerning levels of skewness is the 'old peak' variable.

Visualizing our data by the category that we aim to predict
Below scaterplots show the relationship across our variables per the target category that we aim to predict with our model. These scatterplots show us overall tendency of each variable having a lower or higher value for specific variables, if there is a systematic clustering in our data. We may use this information later during feature selection, model selection, and updating our algorithms.

In [232]:
```
Data_cont_o = pd.concat([Data_cont, data['target']],axis=1)
sns.pairplot(Data_cont_o, kind="scatter", hue="target", plot_kws=dict(s=8
plt.show()
```



Data Cleaning

Check if data has any missing values. As printed below, data includes no missing values, so we do not need to fill any missing values or to exclude any columns and rows.

In [233]:
```
data.isnull().sum()
```

Out[233]:
```
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
```

Find duplicate rows in data. Below line of code, checks for duplicates if in a given row, all columns has exactly the same values. Our data includes one duplicate row, which is dropped from the dataframe.

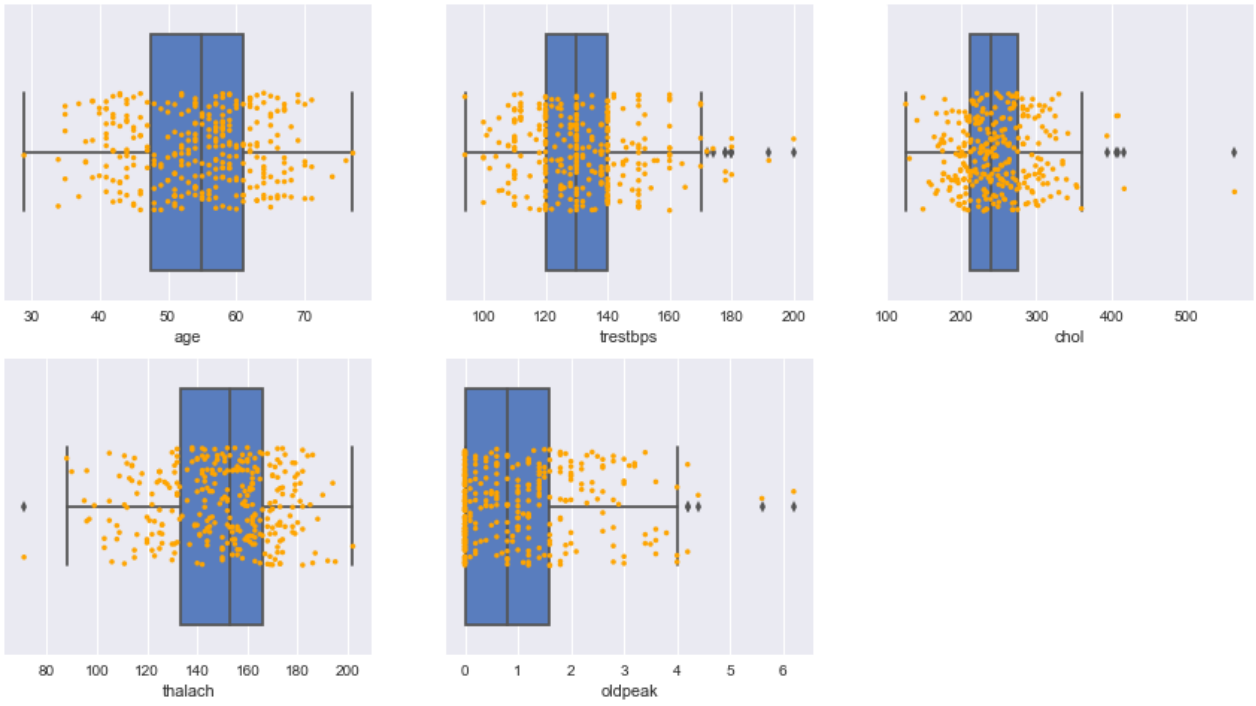
In [234]:
```
duplicated_data = data.duplicated(subset=None, keep='first')
index_duplicate =[i for i, x in enumerate(duplicated_data) if x]
print(data.size)
clean_data = data.drop(index_duplicate)
print(clean_data.size)
```

```
4242
4228
```

Check if data has any outliers. Note that in the below figure, the raw data is also superimposed on the boxplots with the use of jitter. In this context, jitter assures to separate the data so that they aren't plotted directly on top of each other. As an example, without superimposing data on boxplot with jitter, we may have decided that here are 6 possible outliers in that variable, instead of 9. Such biases can affect our decision on how to deal with outliers. A second advantage of jittering is that it allows us to visualize the type of distribution that may possibly be hidden otherwise.

In [235]:
```
fig, axs = plt.subplots(nrows = 3, ncols=2, figsize = (15,8))

plot_number = 1
for i, x in enumerate(Data_cont):
    ax = plt.subplot(2, 3, plot_number)
    sns.boxplot(Data_cont[x])
    sns.stripplot(Data_cont[x], color="orange", jitter=0.2, size=3.5)
    plot_number=plot_number+1
plt.show()
```



How to deal with outliers in the data?

- Things to consider:
- Although, 3 of our continus variables show a number of outliers, these are univariate outliers, meaning they qualify as outliers only for one of the many variables that are used for the model. There are some available methods to identify multivariate outliers as well (i.e. using PCS to identify a cutoff for multivariate outliers).
- Second thing to consider is if these are real outiers. Our samle is composed on 302 unique datapoints and with more data, our distribution might be more close to a normal distribution. Moreover, even if an individual has extreme values, is it really their true measurement or perhaps a measurement error (i.e. for cholesterol variable, is it possible for a person to have a value above 500?)
- Lastly, does it make sense to exclude these outliers from the model estimation perspective. Removal of outliers can perhaps increase the models performance in training (dependent on model selected), but if we train a model without outliers, when thsi model will be used in predicting new coming data with outliers, the performance of our model might decrease.

For now, although we will keep in mind the existance of these outliers, we wil decide what to do with these after model selection and testing. We may perhaps compare model performance with and without outliers in our dataset.

In [ ]: