Clinical Data Explanation and Standard EDA

In this kernel, I'm going to explain the data with some related information about heart disease.

There are two parts in this kernel.
In the first part, I will answer the following two questions:

1. How these fields are related to heart disease?
2. Why these fields can be used to diagnose heart disease?

In the second part, I will give a basic classification solution with 3-layer NN and lightgbm. The best accuracy recorded is *95%* with lightgbm.

Note that the original data descriptions do not correspond to the data, thus I've manually modified them to make them look more realistic.

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
 (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

First, we need to handle null values.
Hopefully there are no null values in this dataset.

In [2]:

```
df = pd.read_csv("../input/heart.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
```

```
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

In [3]:

```
df.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
0	63	1	3	145	233	1	0	150
1	37	1	2	130	250	0	1	187
2	41	0	1	130	204	0	0	172
3	56	1	1	120	236	0	1	178
4	57	0	0	120	354	0	1	163

Here is other fields' correlation with target. Most fields are highly correlated with heart disease.

In [4]:

```
corr = df.corr()['target'].abs().sort_values()
corr
```

Out[4]:

```
fbs         0.028046
chol        0.085239
restecg     0.137230
trestbps    0.144931
age         0.225439
sex         0.280937
thal        0.344029
slope       0.345877
ca          0.391724
thalach     0.421741
oldpeak     0.430696
cp          0.433798
exang       0.436757
target      1.000000
Name: target, dtype: float64
```

Let's plot the differences between people with and without heart diseases side by side.

Fields that are more correlated with heart diseases are put at the bottom.

Part I: Comparison between people with(left) and without(right) heart diseases

In this part, I will answer the following two questions with side-by-side plot of each feature:

1. How these fields are related to heart disease?

2. Why these fields can be used to diagnose heart disease?

In [5]:

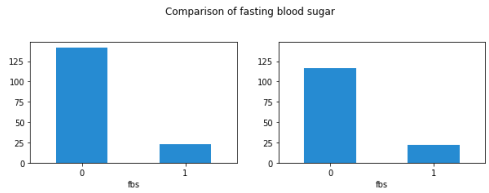
```
# Helper function for plotting side by side
def sideplot(df, col, kind="bar", title=None):
    assert kind in ["bar", "hist"]
    fig = plt.figure(figsize=(10, 6))
    if kind == "bar":
        ax1 = plt.subplot(2, 2, 1)
        df[df.target == 1][['target', col]].groupby(
col).count().plot(kind='bar', rot=0, legend=False, a
x=ax1, color="#268bd2")
        ax2 = plt.subplot(2, 2, 2)
        df[df.target == 0][['target', col]].groupby(
col).count().plot(kind='bar', rot=0, legend=False, a
x=ax2, color="#268bd2")
    else:
        ax1 = plt.subplot(2, 2, 1)
        plt.hist(df[df.target == 1][col], color="#26
8bd2")
        plt.xlabel(col)
        ax2 = plt.subplot(2, 2, 2)
        plt.hist(df[df.target == 0][col], color="#26
8bd2")
        plt.xlabel(col)
    # Re-adjusting
    ylim = (0, max(ax1.get_ylim()[1], ax2.get_ylim()
[1]))
    ax1.set_ylim(ylim)
    ax2.set_ylim(ylim)
    xlim = (min(ax1.get_xlim()[0], ax2.get_xlim()[0
]), max(ax1.get_xlim()[1], ax2.get_xlim()[1]))
    ax1.set_xlim(xlim)
    ax2.set_xlim(xlim)
    if title is not None:
        fig.suptitle(title)
    #plt.subplots_adjust(top=0.99)
```

Fasting blood sugar(FBS)

People with FBS higher than 120 mg/dl are marked as 1, else 0.

In [6]:

```
sideplot(df, "fbs", kind="bar", title="Comparison of
fasting blood sugar")
```



Fasting blood sugar is an important feature for measuring diabetes. Though it is not directly related to heart diseases, It is pointed out that people with diabetes have a higher epidemiological estimate lifetime risk (LTR) of chronical heart disease[1].

Jin C[2] finds out that the trajectories of fasting blood sugar are significantly associated with the risk of myocardio infarction. Since we only have a single value instead of continuous data, it does not look

only have a single value instead of continuous data, it does not look quite useful here to analyse heart disease.

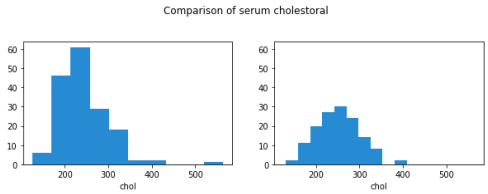
1: Turin TC, Okamura T, Rumana N, Afzal AR, Watanabe M, Higashiyama A, Nakao YM, Nakai M, Takegami M, Nishimura K, Kokubo Y, Okayama A, Miyamoto Y. Diabetes and lifetime risk of coronary heart disease. Prim Care Diabetes. 2017 Oct;11(5):461-466. doi: 10.1016/j.pcd.2017.04.007. Epub 2017 May 22. PubMed PMID: 28545843.

2: Jin C, Chen S, Vaidya A, Wu Y, Wu Z, Hu FB, Kris-Etherton P, Wu S, Gao X. Longitudinal Change in Fasting Blood Glucose and Myocardial Infarction Risk in a Population Without Diabetes. Diabetes Care. 2017 Nov;40(11):1565-1572. doi: 10.2337/dc17-0610. Epub 2017 Sep 8. PubMed PMID: 28887409; PubMed Central PMCID: PMC5652588.

Serum cholestoral(chol)

In [7]:

```
sideplot(df, "chol", kind="hist", title="Comparison
 of serum cholestoral")
```



People who have higher cholesterol in their blood (or hypercholesterolemia) are more likely to have their blood vessels narrowed with more cholesterol (mostly LDL cholesterol) precipited, which is known as atherosclerosis.

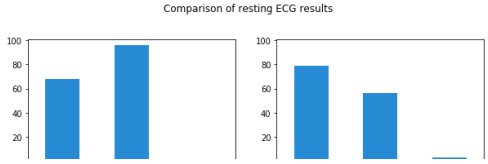
Coronary heart disease, which is one of the most common heart diseases, comes from lack of blood supply for the heart because of atherosclerosis of coronary arteries.

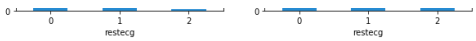
Resting electrocardiographic results (restecg)

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

In [8]:

```
sideplot(df, "restecg", kind="bar", title="Compariso
n of resting ECG results")
```





In ECG, all the three ST-T wave abnormality menstioned(T wave inversion; ST elevation; ST depression of greater than 0.05 mV) are possible signs of myo infarction.
T wave inversion is a sign of lacking blood in endocardium, while ST abnomality indicates there are some damage to the myocardium due to lack of blood for a relatively long period of time.

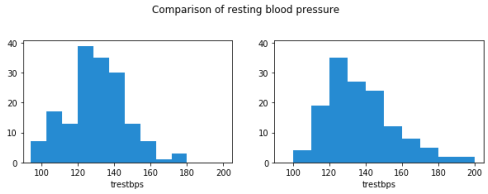
Resting blood pressure (trestbps)

The data seems to be systolic blood pressure.

In most countries, the normal blood pressure is around 120/80. The 2019 standard of high blood pressure (or hypertension) is over 130/80 in America.

In [9]:

```
sideplot(df, "trestbps", kind="hist", title="Compari
son of resting blood pressure")
```

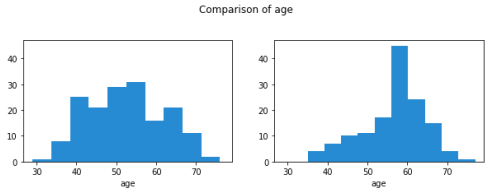


If not controlled, continuous status of high blood pressure can result in congestive heart failure.
In most circumstances, people with high blood pressure also suffered from coronary heart diseases, as high blood pressure can contribute to atherosclerosis.

Age

In [10]:

```
sideplot(df, "age", kind="hist", title="Comparison o
f age")
```

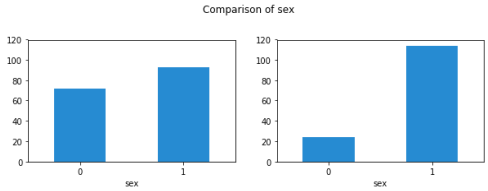


Sex

1 = male; 0 = female

In [11]:

```
sideplot(df, "sex", kind="bar", title="Comparison of
sex")
```



(thal)

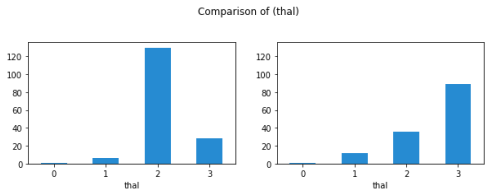
Sorry, I don't know what `thal` means in the dataset. The description shows that it has 3 acceptable values:

- 3 = normal
- 6 = fixed defect
- 7 = reversable defect

But the data here are labeled 0-4. Maybe we should contact the dataset provider.

In [12]:

```
sideplot(df, "thal", kind="bar", title="Comparison o
f (thal)")
```



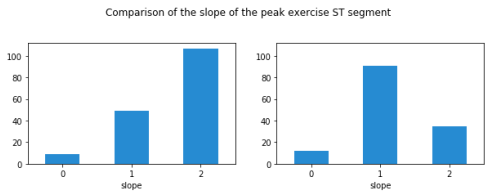
the slope of the peak exercise ST segment (slope)

Adjusted from data description

- Value 0: upsloping
- Value 1: flat
- Value 2: downsloping

In [13]:

```
sideplot(df, "slope", kind="bar", title="Comparison
 of the slope of the peak exercise ST segment")
```

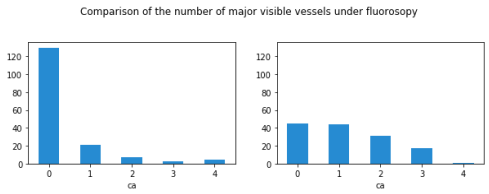


number of major vessels colored by fluorosopy (ca)

The data description says that the number of major vessels ranges from 0 to 3, but actually it ranges from 0 to 4

In [14]:

```
sideplot(df, "ca", kind="bar", title="Comparison of
 the number of major visible vessels under fluorosop
y")
```



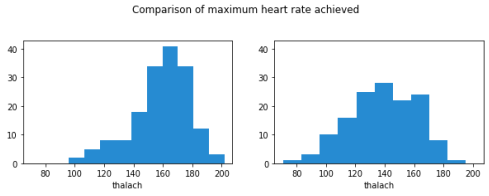
There are two major coronary arteries providing nutrition for the heart. The left one has 3 major branches, and the right one has 5 major branches.

Under fluorosopy, only arteries that have a stable blood transmission can be shown clearly.
Which means the less visible vessels, the worse the heart can receive adequate nutrient and oxygen, and the greater the possibility the patient will suffer a heart attack.

maximum heart rate achieved(thalach)

In [15]:

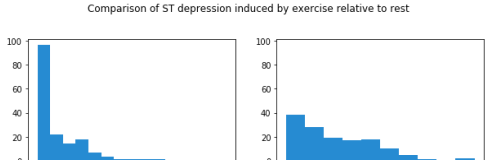
```
sideplot(df, "thalach", kind="hist", title="Comparis
on of maximum heart rate achieved")
```



ST depression induced by exercise relative to rest (oldpeak)

In [16]:

```
sideplot(df, "oldpeak", kind="hist", title="Comparis
on of ST depression induced by exercise relative to
 rest")
```



In exercise, the heart will consume more oxygen and nutrient, which can introduce shortage of blood transmitted to the heart by coronary arteries.

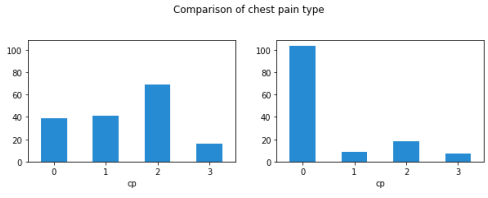
As we learned above, if myocardium lack blood for a long time, the cells in myocardium will be damaged, which can result in depression in ST segment. This feature will expose some not-easy-to-detect heart diseases.

chest pain type (cp)

- Value 0: asymptomatic
- Value 1: typical angina
- Value 2: atypical angina
- Value 3: non-anginal pain

In [17]:

```
sideplot(df, "cp", kind="bar", title="Comparison of
 chest pain type")
```



Present research [1, 2] shows that the number of patients with typical angina are not significantly different from that with atypical angina and non-anginal pain.

It seems that patients with atypical angina and non-anginal pain are more likely to have a heart disease here, which are contradict with the research results, may come from the following reasons:

- The result may be true due to lack of data (we only have 306 records here).
- Indeed atypical angina and non-anginal pain are more likely to have a heart disease.

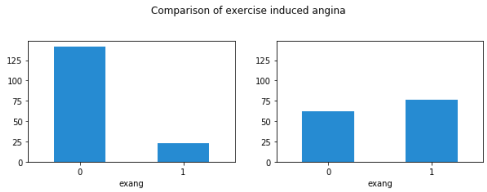
1: Bugaenko VV, Lomakovskiĭ AN. [Alterations in coronary arteries and frequency of episodes of transient myocardial ischemia in patients with ischemic heart disease with typical and atypical angina pectoris]. Lik Sprava. 2002;(2):27-31. Russian. PubMed PMID: 12073253.

2: Hermann LK, Weingart SD, Yoon YM, Genes NG, Nelson BP, Shearer PL, Duvall WL, Henzlova MJ. Comparison of frequency of inducible myocardial ischemia in patients presenting to emergency department with typical versus atypical or nonanginal chest pain. Am J Cardiol. 2010 Jun 1;105(11):1561-4. doi: 10.1016/j.amjcard.2010.01.014. Epub 2010 Apr 10. PubMed PMID: 20494662.

exercise induced angina (exang)

In [18]:

```
sideplot(df, "exang", kind="bar", title="Comparison
 of exercise induced angina")
```



Part II: Basic classification solution

In this part, I am going to give a basic heart disease classification solution with pytorch.

First, let's predict how well the classification model can perform by using linear discriminant analysis (or LDA)
The LDA process similarly as PCA by reducing dimensions of the data, but it tends to make data with different labels most discriminable.

In [19]:

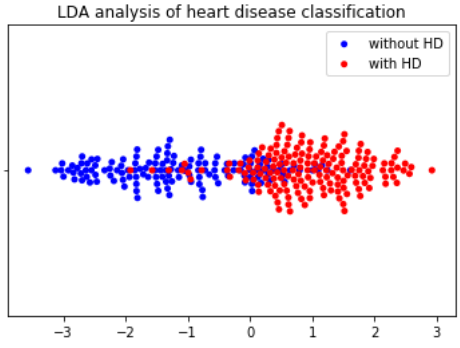
```
from sklearn.discriminant_analysis import LinearDisc
riminantAnalysis as LDA
clf = LDA(n_components=1)

y = df["target"].values
X = clf.fit(df[df.columns[:-1]].values, y).transform
(df[df.columns[:-1]].values)
X = X[:, 0]

sns.swarmplot(X[y == 0], color="b", label="without H
D")
sns.swarmplot(X[y == 1], color="r", label="with HD")
plt.title("LDA analysis of heart disease classificat
ion")
plt.legend()
```

Out[19]:

```
<matplotlib.legend.Legend at 0x7f35f29c
7080>
```



We can see from the LDA graph above that the labelled data are approximately separable.
It seems the model we are going to train will have a comparably good

It seems the model we are going to train will have a comparably good performance.

Preprocessing

NOTE: PREPROCESSING IS CRITICAL TO IMPROVE ACCURACY!!!

Before we train the classifier, we should generally:

1. encode the enum-like features, as non-continuous values should not be considered as continuous ones, since they can have some bad effect on the final result.
2. analyse & drop the outliers.
3. normalize the features to be centered to 0, as some models will be badly affected by that.

First, let's encode the enum-like features.

In [20]:

```
def onehot(ser, num_classes=None):
    """
    One-hot encode the series.
    Example:
    >>> onehot([1, 0, 2], 3)
    array([[0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.]])
    """
    if num_classes == None:
        num_classes = len(np.unique(ser))
    return np.identity(num_classes)[ser]

new_col_names = []
need_encode_col = ["restecg", "thal", "slope", "cp"]
no_encode_col = [col for col in df.columns if col not in need_encode_col]
new_df = df[no_encode_col]
for col in need_encode_col:
    num_classes = len(df[col].unique())
    new_col_names = [f"{col}_{i}" for i in range(num_classes)]
    encoded = pd.DataFrame(onehot(df[col], num_classes), columns=new_col_names, dtype=int)
    new_df = pd.concat([new_df, encoded], axis=1)
new_df.head()
```

Out[20]:

	age	sex	trestbps	chol	fbs	thalach	exang	oldp
0	63	1	145	233	1	150	0	2.3
1	37	1	130	250	0	187	0	3.5
2	41	0	130	204	0	172	0	1.4
3	56	1	120	236	0	178	0	0.8
4	57	0	120	354	0	163	1	0.6

Then we can look at if there are some outliers, for example, with principle component analysis (or PCA).

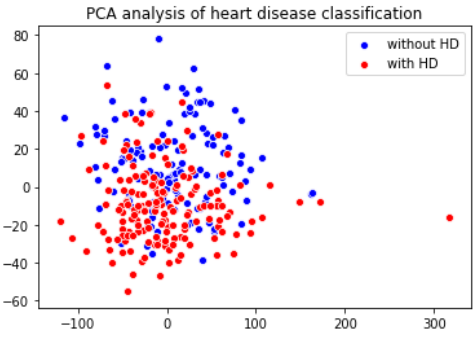
PCA will try to maximize the variance of data.

In [21]:

```
from sklearn.decomposition import PCA
clf = PCA(n_components=2)
data_cols = [col for col in new_df.columns if col !=
"target"]
X = new_df[data_cols]
y = new_df["target"]
X_trans = clf.fit(X, y).transform(X)
sns.scatterplot(X_trans[y == 0][:, 0], X_trans[y ==
0][:, 1], color="b", label="without HD")
sns.scatterplot(X_trans[y == 1][:, 0], X_trans[y ==
1][:, 1], color="r", label="with HD")
plt.title("PCA analysis of heart disease classificat
ion")
plt.legend()
```

Out[21]:

```
<matplotlib.legend.Legend at 0x7f35e805
05f8>
```



We can see a handful of outliers at about (320, -20) and maybe around (150, -10).
We can use sklearn.covariance.EllipticEnvelope for example to drop the outliers if your model will be greatly affected by outliers.

Neural network

First let's train a 3-layer neural network and see what it gave us.

Neural networks are one of the most widely used models today.

In [22]:

```
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

new_df_shfl = shuffle(new_df, random_state=443)
X = new_df_shfl[data_cols].values
y = new_df_shfl["target"].values
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.2, random_state=21)
X_train, X_valid, y_train, y_valid = train_test_spli
t(X, y, test_size=0.25, random_state=80)
```

In [23]:

```
num_epochs = 5000
log_inteval = 250
total_losses = []
total_val_losses = []
lr = 1e-4
lr_decay_inteval = 2500
lr_decay_rate = 0.3
```

In [24]:

```
import torch
from torch import nn, optim
model = nn.Sequential(
    nn.Linear(len(data_cols), 80),
    nn.ReLU(),
    nn.Dropout(0.3),
    nn.Linear(80, 256),
    nn.ReLU(),
    nn.Dropout(0.6),
    nn.Linear(256, 1),
)
loss_fn = torch.nn.BCELoss()
opt = optim.Adam(model.parameters(), lr=lr)

def init_normal(m):
    if type(m) == nn.Linear:
        nn.init.xavier_normal_(m.weight, 0.06)

model.apply(init_normal)
```

Out[24]:

```
Sequential(
  (0): Linear(in_features=23, out_featu
res=80, bias=True)
  (1): ReLU()
  (2): Dropout(p=0.3)
  (3): Linear(in_features=80, out_featu
res=256, bias=True)
  (4): ReLU()
  (5): Dropout(p=0.6)
  (6): Linear(in_features=256, out_feat
ures=1, bias=True)
)
```

In [25]:

```
for epoch in range(1, num_epochs+1):
    y_pred = model(torch.tensor(X_train, dtype=torch
.float))
    y_pred = torch.sigmoid(y_pred)
    opt.zero_grad()
    loss = loss_fn(y_pred[:, 0], torch.tensor(y_trai
n, dtype=torch.float))
    loss.backward()
    opt.step()
    total_losses.append(loss.item())
    if epoch % log_inteval == 0: # Logging
        epochs_ran = epoch
        model.eval()
        with torch.no_grad():
            y_pred = model(torch.tensor(X_test, dtyp
e=torch.float))
            y_pred = torch.sigmoid(y_pred)
            val_loss = loss_fn(y_pred[:, 0], torch.t
ensor(y_test, dtype=torch.float))
            total_val_losses.append(val_loss.item())
        model.train()
```

```
        print(f"total loss in epoch {epoch} = {'%.4f
'%loss}, validation loss = {'%.4f'%val_loss}, lr =
 {'%.2e'%lr}")
        if len(total_val_losses) > 3 and val_loss.it
em() > total_val_losses[-2] and val_loss.item() > to
tal_val_losses[-3]:
            print(f"Validation loss not improving fo
r {log_inteval * 2} epochs, stopping...")
            break
    if epoch % lr_decay_inteval == 0: # Learning rat
e decay
        lr *= lr_decay_rate
        for param_group in opt.param_groups:
            param_group['lr'] = lr
```

```
total loss in epoch 250 = 0.5364, valid
ation loss = 0.5521, lr = 1.00e-04
total loss in epoch 500 = 0.4028, valid
ation loss = 0.4360, lr = 1.00e-04
total loss in epoch 750 = 0.3671, valid
ation loss = 0.3674, lr = 1.00e-04
total loss in epoch 1000 = 0.3010, vali
dation loss = 0.3619, lr = 1.00e-04
total loss in epoch 1250 = 0.2976, vali
dation loss = 0.3551, lr = 1.00e-04
total loss in epoch 1500 = 0.3038, vali
dation loss = 0.3474, lr = 1.00e-04
total loss in epoch 1750 = 0.2837, vali
dation loss = 0.3343, lr = 1.00e-04
total loss in epoch 2000 = 0.2579, vali
dation loss = 0.3127, lr = 1.00e-04
total loss in epoch 2250 = 0.2261, vali
dation loss = 0.2991, lr = 1.00e-04
total loss in epoch 2500 = 0.1947, vali
dation loss = 0.2913, lr = 1.00e-04
total loss in epoch 2750 = 0.2177, vali
dation loss = 0.2846, lr = 3.00e-05
total loss in epoch 3000 = 0.1880, vali
dation loss = 0.2810, lr = 3.00e-05
total loss in epoch 3250 = 0.1764, vali
dation loss = 0.2869, lr = 3.00e-05
```


Data

Data Sources

- Heart Disease UCI
  - heart.csv — 14 columns

Heart Disease UCI

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Last Updated: a year ago (Version 1)

About this Dataset

Context

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.