Heart Disease Data

This is a multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 14 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, oldpeak — ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. This database includes 76 attributes, but all published studies relate to the use of a subset of 14 of them. The Cleveland database is the only one used by ML researchers to date. One of the major tasks on this dataset is to predict based on the given attributes of a patient that whether that particular person has heart disease or not and other is the experimental task to diagnose and find out various insights from this dataset which could help in understanding the problem more.

Hint: heart_disease_uci.csv

Instructions:

1. Use Lifecycle of Data Sciece
2. Use necessary data Preprocess techniques
3. Use various Regression and Classification techniques for comparision
4. Use metrics for regression and classification when needed.
5. Use variosu Pipeline/Hyperparametr tuning techniques for improving performance

```
import numpy as np
import pandas as pd
import pandas_profiling as pp
import math
import random
import seaborn as sns
import matplotlib.pyplot as plt
```

Saved successfully!

```
# preprocessing
import sklearn
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, RobustScaler
from sklearn import metrics
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error, accuracy_score, confusion_matrix, explained_variance_score
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectFromModel, SelectKBest, RFE, chi2

# models
from sklearn.linear_model import LassoCV
from sklearn.svm import LinearSVC
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestClassifier

import warnings
warnings.filterwarnings("ignore")


random_state = 42


data = pd.read_csv("/content/heart_disease_uci.csv")


data.head(3)
```

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
```

```
 0   id        920 non-null    int64
 1   age       920 non-null    int64
 2   sex       920 non-null    object
 3   dataset   920 non-null    object
 4   cp        920 non-null    object
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object
 8   restecg   918 non-null    object
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object
 13  ca        309 non-null    float64
 14  thal      434 non-null    object
 15  num       920 non-null    int64
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

```
data['target'] = data['num']
data = data.drop(columns=['id', 'dataset', 'ca', 'thal', 'num'])
```

```
data = data[data['target'].isin([0, 1])]
data
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	target
0	63	Male	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0
2	67	Male	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	1
3	37	Male	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0
4	41	Female	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0
5	56	Male	atypical angina	120.0	236.0	False	normal	178.0	False	0.8	upsloping	0
				...	...	...	...	...	...	...	...	...
913	62	Male	asymptomatic	158.0	170.0	False	st-t abnormality	138.0	True	0.0	NaN	1
915	54	Female	asymptomatic	127.0	333.0	True	st-t abnormality	154.0	False	0.0	NaN	1
916	62	Male	typical angina	NaN	139.0	False	st-t abnormality	NaN	NaN	NaN	NaN	0
918	58	Male	asymptomatic	NaN	385.0	True	lv hypertrophy	NaN	NaN	NaN	NaN	0
919	62	Male	atypical angina	120.0	254.0	False	lv hypertrophy	93.0	True	0.0	NaN	1

676 rows × 12 columns

Saved successfully!

```
data.describe([.05, .95])
```

	age	trestbps	chol	thalch	oldpeak	target
count	676.000000	643.000000	650.000000	643.000000	640.000000	676.000000
mean	51.715976	131.068429	214.946154	141.838258	0.645938	0.392012
std	9.276611	18.137884	99.125025	25.059654	0.900312	0.488561
min	28.000000	80.000000	0.000000	69.000000	-2.600000	0.000000
5%	36.000000	105.000000	0.000000	98.000000	0.000000	0.000000
50%	53.000000	130.000000	228.500000	143.000000	0.000000	0.000000
95%	66.000000	160.000000	338.550000	180.000000	2.305000	1.000000
max	76.000000	200.000000	603.000000	202.000000	5.000000	1.000000

```
data.describe([.01, .99])
```

	age	trestbps	chol	thalch	oldpeak	target
count	676.000000	643.000000	650.000000	643.000000	640.000000	676.000000
mean	51.715976	131.068429	214.946154	141.838258	0.645938	0.392012
std	9.276611	18.137884	99.125025	25.059654	0.900312	0.488561

```
data = data[(data['chol'] <= 420) & (data['oldpeak'] >=0) & (data['oldpeak'] <=4)].reset_index(drop=True)
data = data.dropna().reset_index(drop=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 362 entries, 0 to 361
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       362 non-null    int64
 1   sex       362 non-null    object
 2   cp        362 non-null    object
 3   trestbps  362 non-null    float64
 4   chol      362 non-null    float64
 5   fbs       362 non-null    object
 6   restecg   362 non-null    object
 7   thalch    362 non-null    float64
 8   exang     362 non-null    object
 9   oldpeak   362 non-null    float64
 10  slope     362 non-null    object
 11  target    362 non-null    int64
dtypes: float64(4), int64(2), object(6)
memory usage: 34.1+ KB
```

```
data.describe()
```

Saved successfully!

	age	trestbps	chol	thalch	oldpeak	target
			362.000000	362.000000	362.000000	362.000000
			230.179558	143.469613	0.984254	0.439227
std	8.803976	17.740539	78.592559	25.301187	0.898136	0.496980
min	29.000000	92.000000	0.000000	82.000000	0.000000	0.000000
25%	47.000000	120.000000	204.000000	124.000000	0.000000	0.000000
50%	54.000000	130.000000	235.500000	147.000000	1.000000	0.000000
75%	59.000000	140.000000	273.000000	162.000000	1.500000	1.000000
max	76.000000	200.000000	417.000000	202.000000	4.000000	1.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 362 entries, 0 to 361
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       362 non-null    int64
 1   sex       362 non-null    object
 2   cp        362 non-null    object
 3   trestbps  362 non-null    float64
 4   chol      362 non-null    float64
 5   fbs       362 non-null    object
 6   restecg   362 non-null    object
 7   thalch    362 non-null    float64
 8   exang     362 non-null    object
 9   oldpeak   362 non-null    float64
 10  slope     362 non-null    object
 11  target    362 non-null    int64
dtypes: float64(4), int64(2), object(6)
memory usage: 34.1+ KB
```

```
def str_features_to_numeric(data):
    # Transforms all string features of the df to numeric features

    # Determination categorical features
    categorical_columns = []
    numerics = ['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']
    features = data.columns.values.tolist()
```

```
    for col in features:
        if data[col].dtype in numerics: continue
        categorical_columns.append(col)

    # Encoding categorical features
    for col in categorical_columns:
        if col in data.columns:
            le = LabelEncoder()
            le.fit(list(data[col].astype(str).values))
            data[col] = le.transform(list(data[col].astype(str).values))

    return data

# Transform all string features of the df to numeric features
data = str_features_to_numeric(data)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 362 entries, 0 to 361
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       362 non-null    int64
 1   sex       362 non-null    int64
 2   cp        362 non-null    int64
 3   trestbps  362 non-null    float64
 4   chol      362 non-null    float64
 5   fbs       362 non-null    int64
 6   restecg   362 non-null    int64
 7   thalch    362 non-null    float64
 8   exang     362 non-null    int64
 9   oldpeak   362 non-null    float64
 10  slope     362 non-null    int64
 11  target    362 non-null    int64
```



```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	
count	362.000000	362.000000	362.000000	362.000000	362.000000	362.000000	362.000000	362.000000	362.000000	362.000000	362.0
mean	53.049724	0.698895	0.895028	132.085635	230.179558	0.132597	0.812155	143.469613	0.419890	0.984254	1.3
std	8.803976	0.459373	1.001399	17.740539	78.592559	0.339608	0.598304	25.301187	0.494224	0.898136	0.5
min	29.000000	0.000000	0.000000	92.000000	0.000000	0.000000	0.000000	82.000000	0.000000	0.000000	0.0
25%	47.000000	0.000000	0.000000	120.000000	204.000000	0.000000	0.000000	124.000000	0.000000	0.000000	1.0
50%	54.000000	1.000000	1.000000	130.000000	235.500000	0.000000	1.000000	147.000000	0.000000	1.000000	1.0
75%	59.000000	1.000000	2.000000	140.000000	273.000000	0.000000	1.000000	162.000000	1.000000	1.500000	2.0
max	76.000000	1.000000	3.000000	200.000000	417.000000	1.000000	2.000000	202.000000	1.000000	4.000000	2.0

```
def fe_creation(df):
    df['age2'] = df['age']//10
    df['trestbps2'] = df['trestbps']//10
    df['chol2'] = df['chol']//60
    df['thalch2'] = df['thalch']//40
    df['oldpeak2'] = df['oldpeak']//0.4
    for i in ['sex', 'age2', 'fbs', 'restecg', 'exang']:
        for j in ['cp','trestbps2', 'chol2', 'thalch2', 'oldpeak2', 'slope']:
            df[i + "_" + j] = df[i].astype('str') + "_" + df[j].astype('str')
    return df

data = fe_creation(data)
```

```
pd.set_option('max_columns', len(data.columns)+1)
len(data.columns)
```

```
47
```

```
# Transform all string features of the df to numeric features
data = str_features_to_numeric(data)
data.head(3)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	target	age2	trestbps2	chol2	thalch2	oldpeak2	sex_cp	sex_t
0	63	1	3	145.0	233.0	1	0	150.0	0	2.3	0	0	6	14.0	3.0	3.0	5.0	7	
1	67	1	0	120.0	229.0	0	0	129.0	1	2.6	1	1	6	12.0	3.0	3.0	6.0	4	
2	37	1	2	130.0	250.0	0	1	187.0	0	3.5	0	0	3	13.0	4.0	4.0	8.0	6	

```
data.shape
```

```
(362, 47)
```

```
train = data.copy()
target = train.pop('target')
train.head(2)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	age2	trestbps2	chol2	thalch2	oldpeak2	sex_cp	sex_trestbps2
0	63	1	3	145.0	233.0	1	0	150.0	0	2.3	0	6	14.0	3.0	3.0	5.0	7	14
1	67	1	0	120.0	229.0	0	0	129.0	1	2.6	1	6	12.0	3.0	3.0	6.0	4	12

Saved successfully!

features that we need to choose as a result
excessive number of features, which we will choose at each stage

```
# Threshold for removing correlated variables
threshold = 0.9

def highlight(value):
    if value > threshold:
        style = 'background-color: black'
    else:
        style = 'background-color: blue'
    return style

# Absolute value correlation matrix
corr_matrix = data.corr().abs().round(2)
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
upper.style.format("{:.2f}").applymap(highlight)
```

thalch2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
oldpeak2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
sex_cp	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
sex_trestbps2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
sex_chol2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
sex_thalch2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
sex_oldpeak2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
sex_slope	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
age2_cp	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
age2_trestbps2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
age2_chol2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
age2_thalch2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
age2_oldpeak2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
age2_slope	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
fbs_cp	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
fbs_trestbps2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
fbs_chol2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
fbs_thalch2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
fbs_oldpeak2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
fbs_slope	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
restecg_cp	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
restecg_thalch2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
restecg_oldpeak2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
restecg_slope	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
exang_cp	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
exang_trestbps2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
exang_chol2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
exang_thalch2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
exang_oldpeak2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
exang_slope	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

Saved successfully!

```
# Select columns with correlations above threshold
collinear_features = [column for column in upper.columns if any(upper[column] > threshold)]
features_filtered = data.drop(columns = collinear_features)
print('The number of features that passed the collinearity threshold: ', features_filtered.shape[1])
features_best.append(features_filtered.columns.tolist())
```

```
The number of features that passed the collinearity threshold:  23
```

```
lsvc = LinearSVC(C=0.1, penalty="l1", dual=False).fit(train, target)
model = SelectFromModel(lsvc, prefit=True)
X_new = model.transform(train)
X_selected_df = pd.DataFrame(X_new, columns=[train.columns[i] for i in range(len(train.columns)) if model.get_support()[i]])
features_best.append(X_selected_df.columns.tolist())
```

```
lasso = LassoCV(cv=3).fit(train, target)
model = SelectFromModel(lasso, prefit=True)
X_new = model.transform(train)
```

```
X_selected_df = pd.DataFrame(X_new, columns=[train.columns[i] for i in range(len(train.columns)) if model.get_support()[i]])
features_best.append(X_selected_df.columns.tolist())

# Visualization from https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
# but to k='all'
bestfeatures = SelectKBest(score_func=chi2, k='all')
fit = bestfeatures.fit(train, target)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(train.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Feature','Score']  #naming the dataframe columns
features_best.append(featureScores.nlargest(num_features_max,'Score')['Feature'].tolist())
print(featureScores.nlargest(len(dfcolumns),'Score'))
```

```
           Feature      Score
44    exang_oldpeak2  465.998249
41   exang_trestbps2  334.796553
7          thalch  313.526041
42      exang_chol2  157.634247
20     sex_oldpeak2  132.273799
17    sex_trestbps2  103.586280
38  restecg_oldpeak2   92.646747
15        oldpeak2   80.815162
2             cp   72.968423
35 restecg_trestbps2   57.716217
4           chol   53.672855
43    exang_thalch2   51.270836
28         fbs_cp   51.166397
8          exang   45.422026
32     fbs_oldpeak2   40.018109
18       sex_chol2   38.775596
9          oldpeak   35.483990
40        exang_cp   34.227582
45      exang_slope   33.239711
26    age2_oldpeak2   11.360552
10          slope   10.515031
1             sex    9.930103
33       fbs_slope    9.679364
14         thalch2    9.434070
22         age2_cp    5.301202
21       sex_slope    4.931790
6          restecg    4.915876
29    fbs_trestbps2    4.433304
23   age2_trestbps2    3.309412
25     age2_thalch2    2.672248
12        trestbps2    2.453725
19      sex_thalch2    1.868808
16         sex_cp    1.634727
30       fbs_chol2    1.348029
39    restecg_slope    1.296849
27       age2_slope    1.072859
0             age    0.951697
13          chol2    0.915487
37   restecg_thalch2    0.614009
5             fbs    0.099183
24       age2_chol2    0.020436
11          age2    0.017083
34       restecg_cp    0.001909
```

Saved successfully!

```
rfe_selector = RFE(estimator=LogisticRegression(), n_features_to_select=num_features_max, step=10, verbose=5)
rfe_selector.fit(train, target)
rfe_support = rfe_selector.get_support()
rfe_feature = train.loc[:,rfe_support].columns.tolist()
print(str(len(rfe_feature)), 'selected features')
```

```
Fitting estimator with 46 features.
Fitting estimator with 36 features.
35 selected features
```

```
embeded_rf_selector = SelectFromModel(RandomForestClassifier(n_estimators=200), threshold='1.25*median')
embeded_rf_selector.fit(train, target)
```

```
SelectFromModel(estimator=RandomForestClassifier(n_estimators=200),
            threshold='1.25*median')
```

```
embeded_rf_support = embeded_rf_selector.get_support()
embeded_rf_feature = train.loc[:,embeded_rf_support].columns.tolist()
print(str(len(embeded_rf_feature)), 'selected features')
```

```
16 selected features
```

```
# Check whether all features have a sufficiently different meaning
selector = VarianceThreshold(threshold=10)
np.shape(selector.fit_transform(data))
features_best.append(list(np.array(data.columns)[selector.get_support(indices=False)]))
```

```
features_best
```

```
[['age',
  'sex',
  'cp',
  'trestbps',
  'chol',
  'fbs',
  'restecg',
  'thalch',
  'exang',
  'oldpeak',
  'slope',
  'target',
  'thalch2',
  'sex_cp',
  'sex_thalch2',
  'sex_oldpeak2',
  'fbs_cp',
  'fbs_trestbps2',
  'fbs_chol2',
  'fbs_thalch2',
  'fbs_oldpeak2',
  'fbs_slope',
```

Saved successfully!

```
  'trestbps',
  'chol',
  'thalch',
  'slope',
  'sex_chol2',
  'sex_oldpeak2',
  'age2_cp',
  'age2_trestbps2',
  'age2_slope',
  'fbs_cp',
  'restecg_thalch2',
  'exang_chol2',
  'exang_oldpeak2'],
 ['age',
  'cp',
  'trestbps',
  'chol',
  'thalch',
  'slope',
  'sex_chol2',
  'sex_oldpeak2',
  'age2_cp',
  'age2_trestbps2',
  'fbs_slope',
  'restecg_slope',
  'exang_chol2',
  'exang_oldpeak2'],
 ['exang_oldpeak2',
  'exang_trestbps2',
  'thalch',
  'exang_chol2',
  'sex_oldpeak2',
  'sex_trestbps2',
```

```
# The element is in at least one list of optimal features
main_cols_max = features_best[0]
for i in range(len(features_best)-1):
    main_cols_max = list(set(main_cols_max) | set(features_best[i+1]))
main_cols_max
```

```
['age2_thalch2',
 'trestbps2',
 'sex_chol2',
```

```
        'thalch',
        'sex_slope',
        'fbs',
        'exang_chol2',
        'sex',
        'restecg_trestbps2',
        'fbs_cp',
        'fbs_slope',
        'restecg_chol2',
        'exang_thalch2',
        'age',
        'fbs_oldpeak2',
        'fbs_thalch2',
        'age2_chol2',
        'target',
        'sex_oldpeak2',
        'age2_oldpeak2',
        'age2_cp',
        'restecg',
        'cp',
        'exang_trestbps2',
        'exang_slope',
        'exang',
        'age2_slope',
        'exang_cp',
        'exang_oldpeak2',
        'restecg_thalch2',
        'fbs_chol2',
        'slope',
        'sex_thalch2',
        'chol',
        'fbs_trestbps2',
        'restecg_slope',
        'thalch2',
        'trestbps',
        'restecg_oldpeak2',
        'sex_trestbps2',
        'oldpeak2',
        'age2_trestbps2']
```

Saved successfully!

```
len(main_cols_max)
```

```
44
```

```
# The element is in all lists of optimal features
main_cols_min = features_best[0]
for i in range(len(features_best)-1):
    main_cols_min = list(set(main_cols_min).intersection(set(features_best[i+1])))
main_cols_min
```

```
['chol', 'thalch', 'sex_oldpeak2', 'trestbps']
```

```
# Most common items in all lists of optimal features
main_cols = []
main_cols_opt = {feature_name : 0 for feature_name in data.columns.tolist()}
for i in range(len(features_best)):
    for feature_name in features_best[i]:
        main_cols_opt[feature_name] += 1
df_main_cols_opt = pd.DataFrame.from_dict(main_cols_opt, orient='index', columns=['Num'])
df_main_cols_opt.sort_values(by=['Num'], ascending=False).head(num_features_opt)
```

	Num
trestbps	5
chol	5
thalch	5
sex_oldpeak2	5
age	4
slope	4
exang_oldpeak2	4
exang_chol2	4
age2_trestbps2	4
sex_chol2	4
age2_cp	4
cp	4
fbs_cp	3
fbs_slope	3
fbs_trestbps2	3
fbs_oldpeak2	3

```
main_cols = df_main_cols_opt.nlargest(num_features_opt, 'Num').index.tolist()
if not 'target' in main_cols:
    main_cols.append('target')
main_cols
```

```
['trestbps',
 'age',
 'cp',
 'slope',
 'sex_chol2',
 'age2_cp',
 'age2_trestbps2',
 'exang_chol2',
 'exang_oldpeak2',
 'fbs_cp',
 'fbs_trestbps2',
 'fbs_oldpeak2',
 'fbs_slope',
 'sex',
 'restecg',
 'exang',
 'oldpeak',
 'thalch2',
 'sex_trestbps2',
 'sex_thalch2',
 'age2_oldpeak2',
 'fbs_thalch2',
 'target']
```

Saved successfully!