Heart Disease prediction

Columns in the Heart Disease dataset

```
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
```

In [1]:

```
# importing the libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
%matplotlib inline
```

In [2]:

```
# Importing machine learning models

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

Heart_Disease Dataset

In [3]:

```
# loading the data

df = pd.read_csv('c://users/santhosh reddy/desktop/untitled folder/untitled folder/heart.cs
```

In [4]:

```
df.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

In [5]:

```
df.tail()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

In [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
# Finding the correlation matrix for the whole dataset

correlation = df.corr()
```

In [8]:

```
# Printing the correlation matric

correlation
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-

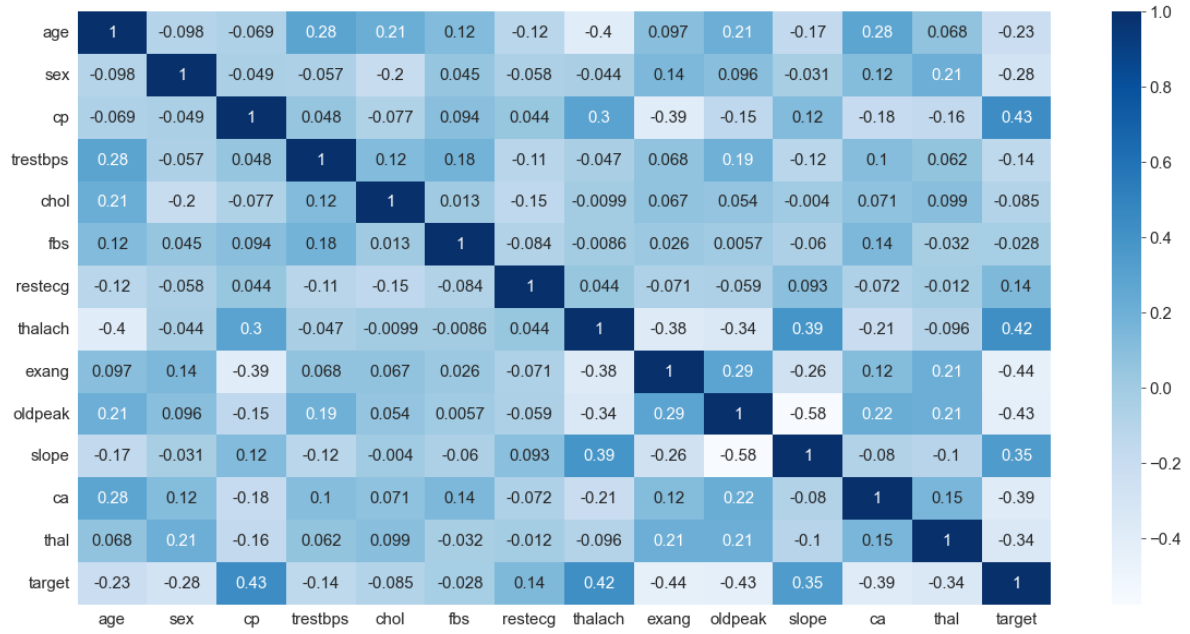
In [9]:

```
# Plotting the correlation matrix as a heatmap

plt.figure(figsize=(20,10))
matplotlib.rcParams['font.size']=15
sns.set_style('whitegrid')
sns.heatmap(correlation, annot=True, cmap='Blues')
```

Out[9]:

```
<AxesSubplot:>
```



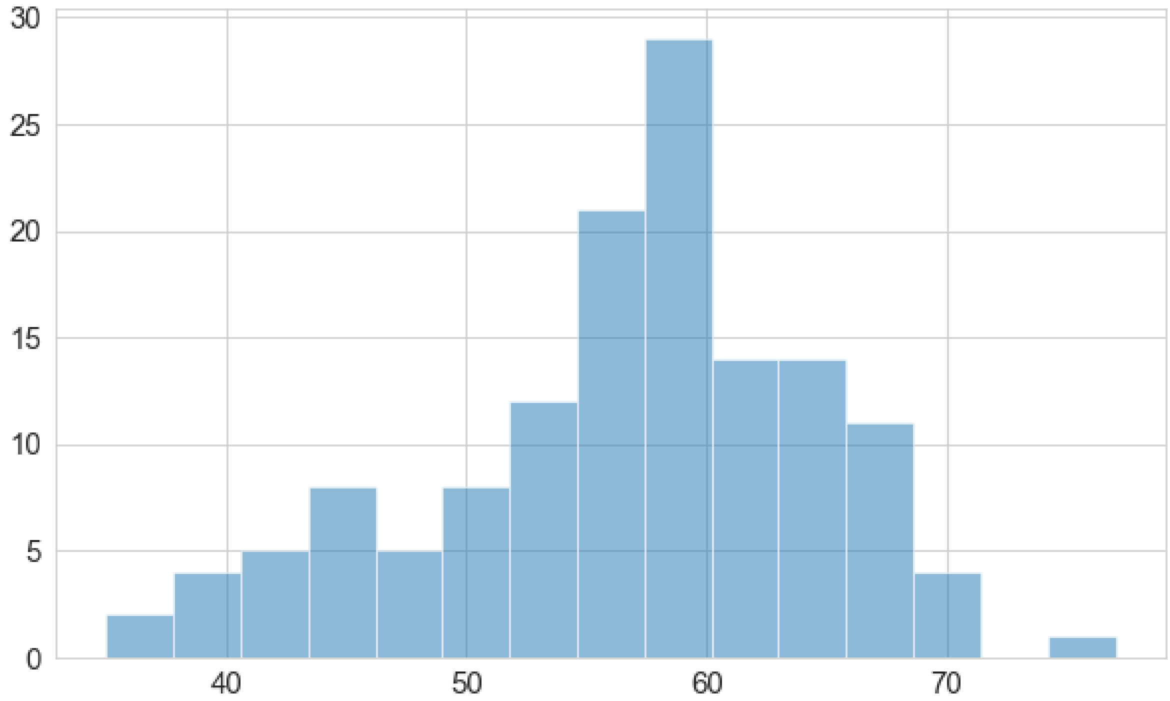
In [10]:

```
# Histogram of age of the heart diseased

plt.figure(figsize=(10,6))
plt.hist(df['age'][df['target']==0], alpha=0.5, bins=15)
```

Out[10]:

```
(array([ 2.,  4.,  5.,  8.,  5.,  8., 12., 21., 29., 14., 14., 11.,  4.,
         0.,  1.]),
 array([35. , 37.8, 40.6, 43.4, 46.2, 49. , 51.8, 54.6, 57.4, 60.2, 63. ,
        65.8, 68.6, 71.4, 74.2, 77. ]),
 <BarContainer object of 15 artists>)
```



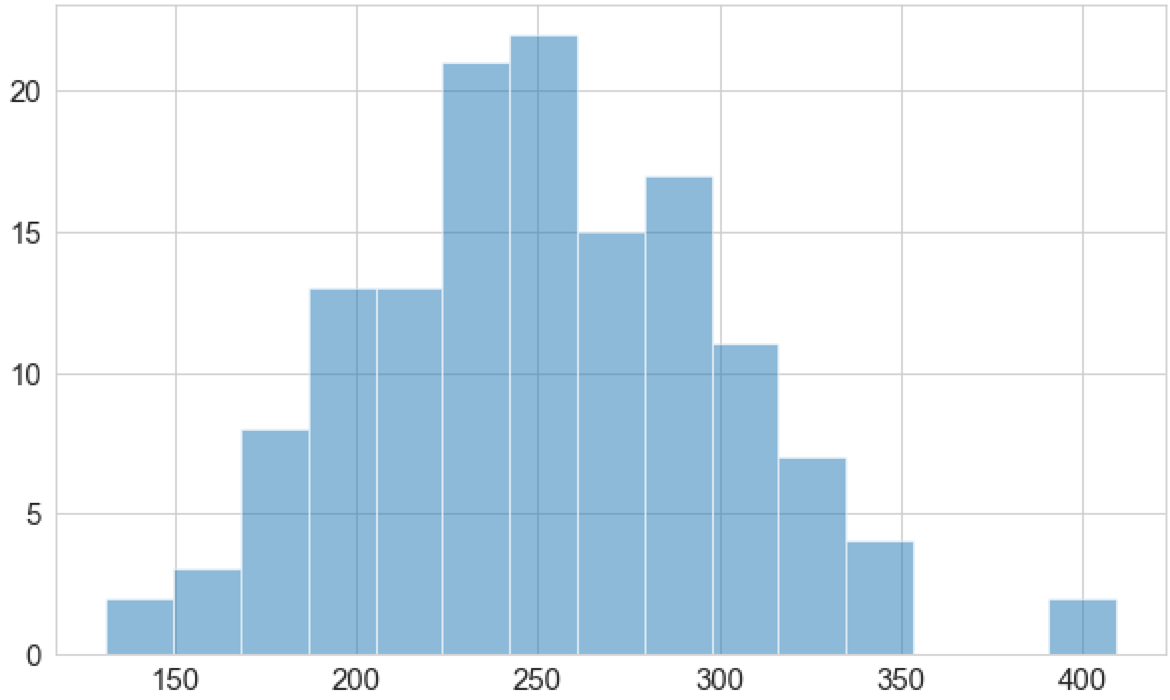
In [11]:

```
# Histogram of age of the heart diseased

plt.figure(figsize=(10,6))
plt.hist(df['chol'][df['target']==0], alpha=0.5, bins=15)
```

Out[11]:

```
(array([ 2.,  3.,  8., 13., 13., 21., 22., 15., 17., 11.,  7.,  4.,  0.,
         0.,  2.]),
 array([131.        , 149.53333333, 168.06666667, 186.6       ,
        205.13333333, 223.66666667, 242.2       , 260.73333333,
        279.26666667, 297.8       , 316.33333333, 334.86666667,
        353.4       , 371.93333333, 390.46666667, 409.        ]),
 <BarContainer object of 15 artists>)
```



In [12]:

```
df.groupby(['sex','target'])['age'].count()
```

Out[12]:

```
sex  target
0    0          24
     1          72
1    0         114
     1          93
Name: age, dtype: int64
```

Females Diseased = 24

Females not-diseased = 72

male diseased = 114

male non-diseased = 93

In [13]:

```
df.shape
```

Out[13]:

```
(303, 14)
```

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
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 [15]:

```
df['target'].value_counts()
```

Out[15]:

```
1    165
0    138
Name: target, dtype: int64
```

0 --> Defective Heart

1 --> Healthy Heart

In [16]:

```
# Splitting the features and target

x = df.drop(columns='target',axis=1)
y = df['target']
```

In [17]:

```
print(x, y)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3   
1     37    1   2       130   250    0        1      187      0      3.5   
2     41    0   1       130   204    0        0      172      0      1.4   
3     56    1   1       120   236    0        1      178      0      0.8   
4     57    0   0       120   354    0        1      163      1      0.6   
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...   
298   57    0   0       140   241    0        1      123      1      0.2   
299   45    1   3       110   264    0        1      132      0      1.2   
300   68    1   0       144   193    1        1      141      0      3.4   
301   57    1   0       130   131    0        1      115      1      1.2   
302   57    0   1       130   236    0        0      174      0      0.0   

     slope  ca  thal  
0        0   0     1  
1        0   0     2  
2        2   0     2  
3        2   0     2  
4        2   0     2  
..     ...  ..   ...  
298      1   0     3  
299      1   0     3  
300      1   2     3  
301      1   1     3  
302      1   1     2  

[303 rows x 13 columns] 0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [18]:

```
x = np.asarray(x)
y = np.asarray(y)
```

Model Selection

In [19]:

```
models = [LogisticRegression(max_iter=10000), SVC(kernel='linear'), KNeighborsClassifier(),
```

In [20]:

```
def compare_models_cross_validation():

    for model in models:

        cv_score = cross_val_score(model, x, y, cv=5)
        mean_accuracy = sum(cv_score)/len(cv_score)
        mean_accuracy = mean_accuracy*100

        print('Cross validation accuracies for the',model,'is',cv_score)
        print('Accuracy score of the',model,'is',round(mean_accuracy,2))
        print('-------------------------------------------------------------------')
```

In [21]:

```
compare_models_cross_validation()
```

```
Cross validation accuracies for the LogisticRegression(max_iter=10000) is
[0.80327869 0.86885246 0.85245902 0.86666667 0.75      ]
Accuracy score of the LogisticRegression(max_iter=10000) is 82.83
-------------------------------------------------------------------
Cross validation accuracies for the SVC(kernel='linear') is [0.81967213 0.88
52459  0.80327869 0.86666667 0.76666667]
Accuracy score of the SVC(kernel='linear') is 82.83
-------------------------------------------------------------------
Cross validation accuracies for the KNeighborsClassifier() is [0.60655738 0.
6557377  0.57377049 0.73333333 0.65      ]
Accuracy score of the KNeighborsClassifier() is 64.39
-------------------------------------------------------------------
Cross validation accuracies for the RandomForestClassifier(random_state=0) i
s [0.85245902 0.90163934 0.81967213 0.81666667 0.8       ]
Accuracy score of the RandomForestClassifier(random_state=0) is 83.81
-------------------------------------------------------------------
```

INFERENCE :
for the Heart Disease dataset, RANDOM FOREST CLASSIFIER has the Highest accuracy value with default Hyperparameters

GridSearchCV

2. Comparing the models with different Hyperparameter values using GridSearchCV

In [22]:

```
model_list = [LogisticRegression(max_iter=10000), SVC(), KNeighborsClassifier(), RandomFore
```

In [23]:

```
# Creating a Dictionary containing Hyperparameters

model_hyperparameters = {

    'log_reg_hyperparameters' : {
        'C' : [1, 5, 10, 20]
    },

    'svc_hyperparameters' : {
        'kernel' : ['linear','poly','rbf','sigmoid'],
        'C' : [1, 5, 10, 20]
    },

    'KNN_hyperparameters' : {
        'n_neighbors' : [3, 5, 10]
    },

    'random_forest_hyperparameters' : {
        'n_estimators' : [10, 20, 50, 100]
    }
}
```

In [24]:

```
print(model_hyperparameters.keys())
```

```
dict_keys(['log_reg_hyperparameters', 'svc_hyperparameters', 'KNN_hyperparam
eters', 'random_forest_hyperparameters'])
```

In [25]:

```
print(model_hyperparameters.values())
```

```
dict_values([{'C': [1, 5, 10, 20]}, {'kernel': ['linear', 'poly', 'rbf', 'si
gmoid'], 'C': [1, 5, 10, 20]}, {'n_neighbors': [3, 5, 10]}, {'n_estimators':
[10, 20, 50, 100]}])
```

In [26]:

```
model_keys = list(model_hyperparameters.keys())
print(model_keys)
```

```
['log_reg_hyperparameters', 'svc_hyperparameters', 'KNN_hyperparameters', 'r
andom_forest_hyperparameters']
```

In [27]:

```
print(model_hyperparameters[model_keys[0]]) # 0 -- log_reg_hyperparameters
model_hyperparameters[model_keys[1]] # 1 -- svc_hyperparameters
```

```
{'C': [1, 5, 10, 20]}
```

Out[27]:

```
{'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [1, 5, 10, 20]}
```

Applying the GridSearchCV

In [28]:

```
def ModelSelection(list_of_models, hyperparameters_dictionary):

    result = []

    i = 0

    for model in list_of_models:
        key = model_keys[i]
        params = hyperparameters_dictionary[key]

        i += 1
        print(model)
        print(params)

        classifier = GridSearchCV(model, params, cv=5)

        # Fitting the model
        classifier.fit(x, y)

        result.append({
            'model used' : model,
            'highest score' : classifier.best_score_,
            'best hyperparameters' : classifier.best_params_

        })
    result_dataframe = pd.DataFrame(result, columns=['model used','highest score','best hyp

    return result_dataframe
```

In [29]:

```
ModelSelection(model_list, model_hyperparameters)
```

```
LogisticRegression(max_iter=10000)
{'C': [1, 5, 10, 20]}
SVC()
{'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [1, 5, 10, 20]}
KNeighborsClassifier()
{'n_neighbors': [3, 5, 10]}
RandomForestClassifier(random_state=0)
{'n_estimators': [10, 20, 50, 100]}
```

Out[29]:

	model used	highest score	best hyperparameters
0	LogisticRegression(max_iter=10000)	0.831585	{'C': 5}
1	SVC()	0.828306	{'C': 1, 'kernel': 'linear'}
2	KNeighborsClassifier()	0.643880	{'n_neighbors': 5}
3	RandomForestClassifier(random_state=0)	0.838087	{'n_estimators': 100}

Random Forest classifier with n_estimators = 100, has the Highest Accuracy score

In [30]:

```
# Spplitting the data into train and test data

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=10, test_size=0.2)
```

In [31]:

```
model = RandomForestClassifier(n_estimators=100)
```

In [32]:

```
model.fit(x_train, y_train)
```

Out[32]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [33]:

```
# Prediction on the train data

train_pred = model.predict(x_train)
```

In [34]:

```
# Accuracy score of the training data

train_accuracy = accuracy_score(train_pred, y_train)
print(train_accuracy)
```

```
1.0
```

In [35]:

```
# Predction on the test data

test_pred = model.predict(x_test)
```

In [36]:

```
# accuracy score of the test data

test_accuracy = accuracy_score(test_pred, y_test)
print(test_accuracy)
```

```
0.7868852459016393
```

Making a predictive system

In [37]:

```
def heart_disease_prediction(input_data):
    # Taing the input data from the user
    input_data = input_data

    # converting the input data into numpy array
    input_data = np.asarray(input_data)

    # Reshaping the input data
    reshaped_data = input_data.reshape(1,-1)

    # Predicting the input data
    predict = model.predict(reshaped_data)

    # Printing diseased or not from the prediction
    if predict == 0:
        print('Diseased')
    elif predict == 1:
        print('Not-Diseased')
```

In [38]:

```
# Creating feature list
features = []

# Taking n inputs from the user and adding them to list
for i in range(13):
    a = input()
    features.append(a)

# printing the features list
print(features)

# calling the heart_disease_prediction function
heart_disease_prediction(features)
```

```
37
1
2
130
250
0
1
187
0
3.5
0
0
2
['37', '1', '2', '130', '250', '0', '1', '187', '0', '3.5', '0', '0', '2']
Not-Diseased
```