Support Vector Machine (SVM)

Source:

https://www.kaggle.com/ronitf/heart-disease-uci (https://www.kaggle.com/ronitf/heart-disease-uci)

Defining the Problem Statement

This dataset records the attributes of a group of patients and whether they have heart disease. From this dataset, we would like to be able to predict the presence of heart disease in patients.

Collecting the Data

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import time
```

In [2]:

```
df=pd.read_csv('heart.csv')
df.head()
```

Out[2]:

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

Modelling

In [3]:

```
df_not = df[df['target']==1]
df_yes = df[df['target']==0]
df_yes.head()
```

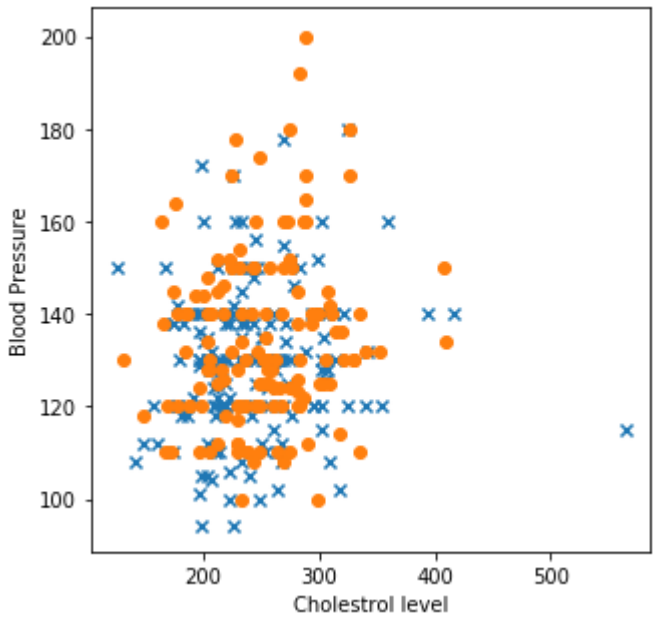
Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
165	67	1	0	160	286	0	0	108	1	1.5	1	3	2	
166	67	1	0	120	229	0	0	129	1	2.6	1	2	3	
167	62	0	0	140	268	0	0	160	0	3.6	0	2	2	
168	63	1	0	130	254	0	0	147	0	1.4	1	1	3	
169	53	1	0	140	203	1	0	155	1	3.1	0	0	3	

We select **cholestrol level** and **blood pressure** as our first set of independent variables since these two factors are the usual prime suspects behind heart disease

In [4]:

```
fig,ax=plt.subplots(figsize=(5,5))
ax.scatter(df_not['chol'],df_not['trestbps'],marker='x')
ax.scatter(df_yes['chol'],df_yes['trestbps'],marker='o')
ax.set(xlabel='Cholestrol level', ylabel='Blood Pressure')
plt.show()
```



The scatter plot of the cholestrol level/blood pressure features suggests the hyperplane may not be that clear. We continue with this test but do not expect a high accuracy score for this model.

In [5]:

```
from sklearn.model_selection import train_test_split

X=df[['chol','trestbps']]
y=df['target']
```

In [6]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [7]:

```
from sklearn.svm import SVC
start=time.time()
svClassifier = SVC(kernel='linear',gamma='scale')
svClassifier.fit(X_train,y_train)
end=time.time()
print(end-start)
```

```
0.030953168869018555
```

In [8]:

```
y_pred=svClassifier.predict(X_test)
y_pred
```

Out[8]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [9]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[9]:

```
0.5394736842105263
```

The accuracy score of cholestrol/blood pressure features is only **54%**.

Modelling Part 2

We try to perform some feature engineering here to pick the best features that would increase the accuracy score.

In [10]:

```
df.corr()
```

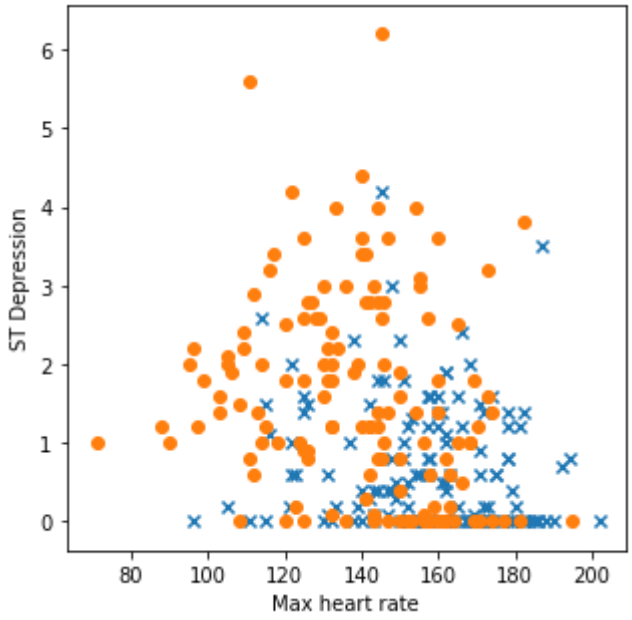
Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalac
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.39852
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.04402
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.29576
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.04669
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.00994
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.00856
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.04412
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.00000
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.37881
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.34418
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.38678
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.21317
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.09643
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.42174

Based on the features correlation table above, **thalach** (maximum heart rate) and **oldpeak** (ST Depression:heart readings after excercise - higher score means higher risk). So we select these two features to test next.

In [11]:

```
fig,ax=plt.subplots(figsize=(5,5))
ax.scatter(df_not['thalach'],df_not['oldpeak'],marker='x')
ax.scatter(df_yes['thalach'],df_yes['oldpeak'],marker='o')
ax.set(xlabel='Max heart rate', ylabel='ST Depression')
plt.show()
```



In [12]:

```
from sklearn.model_selection import train_test_split

X=df[['thalach','oldpeak']]
y=df['target']
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [14]:

```
from sklearn.svm import SVC

start=time.time()
svClassifier = SVC(kernel='poly',gamma='scale')
svClassifier.fit(X_train,y_train)
end=time.time()
print(end-start)
```

```
0.003987550735473633
```

In [15]:

```
y_pred=svClassifier.predict(X_test)
y_pred
```

Out[15]:

```
array([0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1], dtype=int64)
```

In [16]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[16]:

```
0.7894736842105263
```

We have improved the accuracy score from **53% to 79%** without increasing training-duration by selecting features based on the correlation factors. We next consider a multiple features to the model and test the impact on training-duration.

Modelling Part 3

We add a 3rd feature (cp:chest pain) to our previous model. We choose cp because it has a relatively high correlation factor of 0.43 with target.

In [17]:

```
from sklearn.model_selection import train_test_split

X=df[['thalach','oldpeak','cp']]
y=df['target']
```

In [18]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [19]:

```
from sklearn.svm import SVC

start=time.time()
svClassifier = SVC(kernel='linear',gamma='scale')
svClassifier.fit(X_train,y_train)
end=time.time()
print(end-start)
```

```
0.021938085556030273
```

In [20]:

```
y_pred=svClassifier.predict(X_test)
y_pred
```

Out[20]:

```
array([0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1], dtype=int64)
```

In [21]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[21]:

```
0.8157894736842105
```

We improved accuracy score from **78.9% to 81.5%**, while training-duration stayed the same.

Modelling Part 4

We now try adding a fourth feature, called exang. Exang basically measures whether excercise causes angina.

In [22]:

```
from sklearn.model_selection import train_test_split

X=df[['thalach','oldpeak','cp','exang']]
y=df['target']
```

In [23]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [24]:

```
from sklearn.svm import SVC
start=time.time()
svClassifier = SVC(kernel='linear',gamma='scale')
svClassifier.fit(X_train,y_train)
end=time.time()
print(end-start)
```

```
0.021941661834716797
```

In [25]:

```
y_pred=svClassifier.predict(X_test)
y_pred
```

Out[25]:

```
array([0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1], dtype=int64)
```

In [26]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[26]:

```
0.868421052631579
```

We have improved accuracy score from **81.5% to 86.8%**, while training duration stayed the same.

Conclusion

Further test below shows that adding a 5th feature does not improve accuracy score, so we conclude the model with 4 features is the optimal model for predicting heart disease in a patient.

We will fine-tune the 4-feature model by lowering C from its default setting (10) to 0.1. This raises our final accuracy score of the SVM model from **86.8% to 88.2%**.

In [27]:

```
def svmScore(c = 1, gam = 'scale', ker ='rbf'):
    svMachine = SVC(C = c, kernel = ker, gamma = gam)
    svMachine.fit(X_train, y_train)
    y_pred = svMachine.predict(X_test)
    print(accuracy_score(y_test, y_pred))
```

In [28]:

```
kernels = ['linear', 'rbf', 'poly']
for k in kernels:
    svmScore(ker = k)
```

```
0.868421052631579
0.7105263157894737
0.7763157894736842
```

In [29]:

```
svmScore(c = 0.1, ker = 'linear')
svmScore(c = 10, ker = 'linear')
svmScore(c = 100, ker = 'linear')
```

```
0.881578947368421
0.868421052631579
0.8552631578947368
```

Appendix

Test addition of 5th feature

In [30]:

```
from sklearn.model_selection import train_test_split

X=df[['thalach','oldpeak','cp','exang','ca']]
y=df['target']
```

In [31]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [32]:

```
from sklearn.svm import SVC
start=time.time()
svClassifier = SVC(kernel='linear',gamma='scale')
svClassifier.fit(X_train,y_train)
end=time.time()
print(end-start)
```

```
0.016953229904174805
```

In [33]:

```
y_pred=svClassifier.predict(X_test)
y_pred
```

Out[33]:

```
array([0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1], dtype=int64)
```

In [34]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[34]:

```
0.868421052631579
```

In [35]:

```
kernels = ['linear', 'rbf', 'poly']
for k in kernels:
    svmScore(ker = k)
```

```
0.868421052631579
0.7105263157894737
0.7763157894736842
```