Heart Disease Prediction

In another notebook, I performed EDA. In this notebook, I evaluate different models performance in predicting heart disease.

The dataset is available at: [https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset]

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [39]:
```
data = pd.read_csv('heart.csv')
```

In [14]:
```
data.head()
```

Out[14]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	

In [20]:
```
#data.isnull().sum()
```

In [6]:
```
data.duplicated().sum()
```

Out[6]:
```
723
```

In [41]:
```
data = data.drop_duplicates()
data.shape
```

Out[41]:
```
(302, 14)
```

This dataset contains both categorical and numerical features. Categorical variables need to be encoded and numerical variables need to be normalized.

Normalization is only required for models that calculate distances between points.

In [43]:
```
cat_va = []
num_va = []

for column in data.columns:
```

```
    if data[column].nunique() <= 10:
        cat_va.append(column)
    else:
        num_va.append(column)
```

In [73]:
```
print(cat_va)
print(num_va)
```

```
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

Encoding Categorical Variables

'Sex' and 'target' columns are already encoded; as they come with values of 0 and 1. But all the columns whose name is in cat_va need to be encoded.

In [45]:
```
cat_va.remove('sex')
cat_va.remove('target')
```

In [77]:
```
cat_va
```

Out[77]:
```
['cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
```

In [47]:
```
data=pd.get_dummies(data, columns=cat_va, drop_first=True)
```

In [83]:
```
data.head()
```

Out[83]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slop
0	52	1	125	212	168	1.0	0	False	False	False	...	False	F
1	53	1	140	203	155	3.1	0	False	False	False	...	True	F
2	70	1	145	174	125	2.6	0	False	False	False	...	True	F
3	61	1	148	203	161	0.0	0	False	False	False	...	False	F
4	62	0	138	294	106	1.9	0	False	False	False	...	False	

5 rows × 23 columns

Feature Scaling

In [13]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [49]:
```
data[num_va] = scaler.fit_transform(data[num_va])
```

In [95]:
```
data.head(3)
```

Out[95]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...
0	-0.267966	1	-0.376556	-0.667728	0.806035	-0.037124	0	False	False	False	...
1	-0.157260	1	0.478910	-0.841918	0.237495	1.773958	0	False	False	False	...
2	1.724733	1	0.764066	-1.403197	-1.074521	1.342748	0	False	False	False	...

3 rows × 23 columns

In [51]:
```
X = data.drop('target', axis=1)
y = data['target']
```

In [53]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

Modeling

1. Logistic Regression

In [55]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

LR = LogisticRegression()
LR.fit(X_train, y_train)
LR_y = LR.predict(X_test)
LR_accuracy = accuracy_score(y_test, LR_y)
print(LR_accuracy)
```

```
0.7868852459016393
```

2. Support Vector Classifier

In [57]:
```
from sklearn.svm import SVC

svm = SVC(kernel='rbf')
svm.fit(X_train, y_train)
svm_y = svm.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_y)
print(svm_accuracy)
```

```
0.8032786885245902
```

3. KNeighbors Classifier

In [31]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [59]:
```
#determine best k value:

score=[]
for k in range (1, 11):
```

```
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred=knn.predict(X_test)
    score.append(accuracy_score(y_test, y_pred))

score
```

Out[59]:
```
[0.7213114754098361,
 0.8032786885245902,
 0.7049180327868853,
 0.7049180327868853,
 0.7377049180327869,
 0.8032786885245902,
 0.7868852459016393,
 0.8032786885245902,
 0.7704918032786885,
 0.7540983606557377]
```

k = 2 gives best accuracy for knn.

In [61]:
```
knn=KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train,y_train)
knn_y=knn.predict(X_test)
knn_accuracy= accuracy_score(y_test,knn_y)
```

Non_linear ML

No need for encoding or normalization!

In [63]:
```
data = pd.read_csv('heart.csv')
```

In [65]:
```
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.2,
                                                random_state=42)
```

4. Decision Tree Classifier

In [67]:
```
from sklearn.tree import DecisionTreeClassifier


DT = DecisionTreeClassifier()

DT.fit(X_train,y_train)
DT_y= DT.predict(X_test)
DT_accuracy = accuracy_score(y_test,DT_y)
DT_accuracy
```

Out[67]:
```
0.7377049180327869
```

5. Random Forest Classifier

In [69]:
```
from sklearn.ensemble import RandomForestClassifier

RF =RandomForestClassifier()
```

```
RF.fit(X_train, y_train)
RF_y = RF.predict(X_test)
RF_accuracy = accuracy_score(y_test, RF_y)
RF_accuracy
```

Out[69]: 0.7540983606557377

In [71]:
```
final_data = pd.DataFrame({'Models':['LR','SVM','KNN','DT','RF'],
                           'ACC':[LR_accuracy*100,
                                  svm_accuracy*100,
                                  knn_accuracy*100,
                                  DT_accuracy*100,
                                  RF_accuracy*100]})
```

In [73]:
```
final_data
```

Out[73]:

	Models	ACC
0	LR	78.688525
1	SVM	80.327869
2	KNN	80.327869
3	DT	73.770492
4	RF	75.409836

SVM and KNN performed best with this dataset.

Let's see if selecting features with the highest correlation with the target improves models performance.

View Correlation Matrix

In [ ]:
```

```

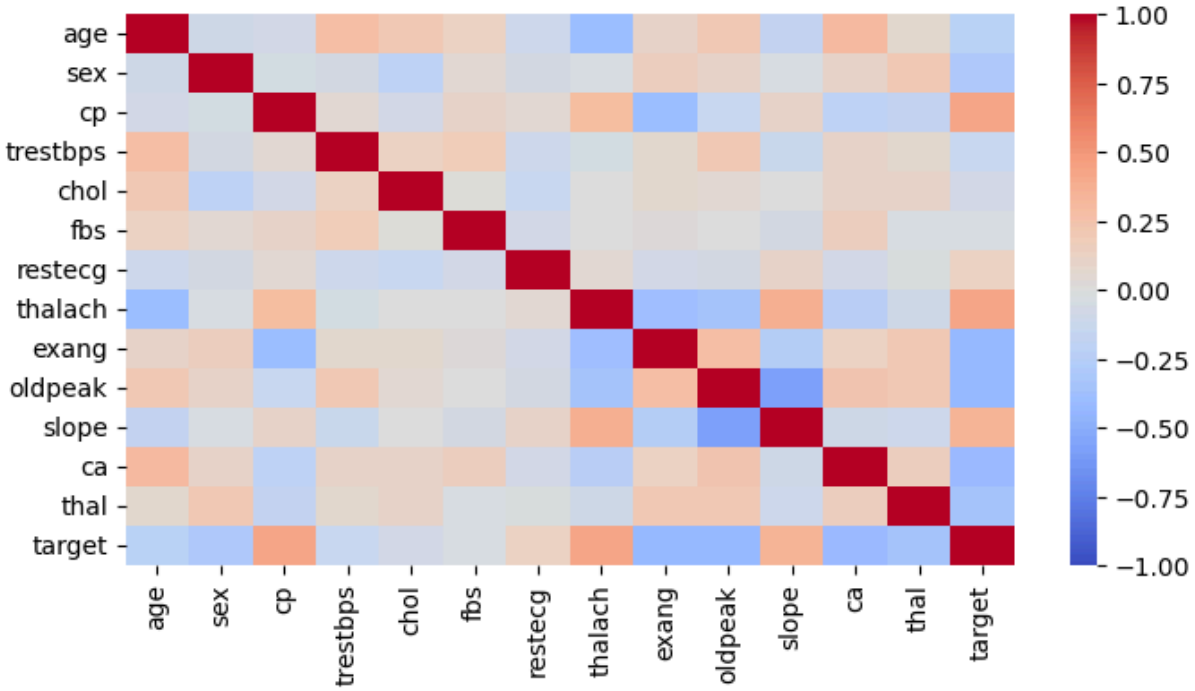
In [ ]:
```

```

In [ ]:
```

```

In [12]:
```
plt.figure(figsize=(8,4))
sns.heatmap(data.corr(), cmap='coolwarm', vmin=-1, vmax=1)
plt.show()
```



Insights from Correlation Matrix

- Chest pain, maximum heart rate, and slope have high correlation with the target.
- Exercise induced angina, oldpeak, ca, and thal negatively correlate wiht the target.

In [21]:
```
data.columns
```

Out[21]:
```
Index(['age', 'sex', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target',
       'cp_1', 'cp_2', 'cp_3', 'fbs_1', 'restecg_1', 'restecg_2', 'exang_1',
       'slope_1', 'slope_2', 'ca_1', 'ca_2', 'ca_3', 'ca_4', 'thal_1',
       'thal_2', 'thal_3'],
      dtype='object')
```

In [23]:
```
X =data[['thalach', 'oldpeak',
       'cp_1', 'cp_2', 'cp_3',
       'slope_1', 'slope_2', 'ca_1', 'ca_2', 'ca_3', 'ca_4', 'thal_1',
       'thal_2', 'thal_3']]
y = data['target']
```

In [25]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

LR_2 = LogisticRegression()
LR_2.fit(X_train, y_train)
LR2_y = LR_2.predict(X_test)
LR2_accuracy = accuracy_score(y_test, LR2_y)
print(LR2_accuracy)
```

```
0.819672131147541
```

In [29]:
```
from sklearn.svm import SVC

svm2 = SVC(kernel='rbf')
svm2.fit(X_train, y_train)
svm2_y = svm2.predict(X_test)
svm2_accuracy = accuracy_score(y_test, svm2_y)
print(svm2_accuracy)
```

```
0.7377049180327869
```

In [33]:
```
score=[]
for k in range (1, 11):
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred=knn.predict(X_test)
    score.append(accuracy_score(y_test, y_pred))

score
```

Out[33]:
```
[0.7213114754098361,
 0.7704918032786885,
 0.7049180327868853,
 0.7540983606557377,
 0.7377049180327869,
 0.7540983606557377,
 0.6885245901639344,
 0.7540983606557377,
 0.7868852459016393,
 0.7868852459016393]
```

In [35]:
```
from sklearn.tree import DecisionTreeClassifier


DT = DecisionTreeClassifier()

DT.fit(X_train,y_train)
DT_y= DT.predict(X_test)
DT_accuracy = accuracy_score(y_test,DT_y)
DT_accuracy
```

Out[35]:
```
0.7377049180327869
```

In [37]:
```
from sklearn.ensemble import RandomForestClassifier

RF =RandomForestClassifier()
RF.fit(X_train, y_train)
RF_y = RF.predict(X_test)
RF_accuracy = accuracy_score(y_test, RF_y)
RF_accuracy
```

Out[37]:
```
0.7213114754098361
```

In [ ]:
```

```