Heart Disease Prediction - likitha

In [28]:
```
import numpy as np
import pandas as pd
```

In [29]:
```
dataset = pd.read_csv('heart.csv')
```

In [30]:
```
dataset.head()
```

Out[30]:

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

In [31]:
```
dataset.columns
```

Out[31]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalac
h',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [32]:
```
categorical = ['sex', 'cp', 'restecg', 'slope', 'thal']
do_not_touch = ['fbs', 'exang']
non_categorical = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca']
```

OneHotEncoding categorical values

In [33]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [34]:
```
ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),categorical)
X = ct.fit_transform(dataset[categorical+do_not_touch+non_categorical])
y = dataset['target'].values
```

In [35]:
```
X[0,:]
```

Out[35]:
```
array([  0. ,   1. ,   0. ,   0. ,   0. ,   1. ,   1. ,   0. ,   0. ,
         1. ,   0. ,   0. ,   0. ,   1. ,   0. ,   0. ,   1. ,   0. ,
        63. , 145. , 233. , 150. ,   2.3,   0. ])
```

In [36]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.1,random_st
```

Last 6 columns of the dataset are non categorical values. So they need to be scaled.

In [37]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [38]:
```
X_train[:,-6:] = scaler.fit_transform(X_train[:,-6:])
X_test[:,-6:] = scaler.transform(X_test[:,-6:])
```

In [39]:
```
X_train[0,:]
```

Out[39]:
```
array([ 0.        ,  1.        ,  0.        ,  1.        ,  0.        ,
        0.        ,  0.        ,  1.        ,  0.        ,  0.        ,
        0.        ,  1.        ,  0.        ,  0.        ,  1.        ,
        0.        ,  0.        ,  0.        , -0.61990074, -0.08877873,
        0.37511601,  0.91545786, -0.37805012, -0.70686683])
```

In [40]:
```
from sklearn.svm import SVC
estimator = SVC()

parameters = [{'kernel':['rbf'],
               'C':[1,10,100,1000],
               'gamma':[1,0.1,0.001,0.0001],
              },
              {'kernel':['poly'],
               'C':[1,10,100,1000],
               'gamma':[1,0.1,0.001,0.0001],
               'degree':range(1,5)}
               ]
```

Using Grid Search to find best fit SVC model

In [41]:
```
from sklearn.model_selection import GridSearchCV
```

In [42]:
```
grid_search = GridSearchCV(
    estimator=estimator,
    param_grid=parameters,
    scoring = 'accuracy',
    n_jobs = 10,
    cv = 10,
    verbose=True
)
```

In [43]:
```
grid_search.fit(X_train, y_train)
grid_search.best_estimator_
```

```
Fitting 10 folds for each of 80 candidates, totalling 800 fits
```

Out[43]:

SVC

SVC(C=100, gamma=0.0001)

In [44]:
```
y_pred = grid_search.best_estimator_.predict(X_test)
```

In [45]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test,y_pred))
accuracy_score(y_test,y_pred)
```

```
[[13  3]
 [ 0 15]]
```

Out[45]:
```
0.9032258064516129
```

So, this is the final confusion matrix and the accuracy score

In [46]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load your dataset from a local CSV file
data = pd.read_csv('heart.csv')  # Replace 'your_dataset.csv' with your file

# Assuming the last column contains the target variable, and the rest are fe
  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the classifier to the training data
rf_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.82
```

In [47]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load your dataset from a local CSV file
data = pd.read_csv('heart.csv')  # Replace 'your_dataset.csv' with your file

# Assuming the last column contains the target variable, and the rest are fe

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar

# Create a Gaussian Naive Bayes classifier
nb_classifier = GaussianNB()

# Fit the classifier to the training data
nb_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = nb_classifier.predict(X_test)g \

\0

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.84
```

In [ ]: