In [1]:

```
import sklearn
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import scipy
import statistics
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import label_binarize
import os
```

In [2]:

```
path = os.getcwd()
iris_df = pd.read_csv(path+'\\Learn Dataset\\iris_dataset_missing.csv')
iris_df_nona = iris_df.dropna()
iris_df_nona["Class"] = list(iris_df_nona.loc[:,"species"].values)
iris_df_nona["Class"]=iris_df_nona["Class"].replace("Iris-versicolor",0).replace("Iris-setosa",1).replace("Iris-virginica",2)

heart_df = pd.read_csv(path+'\\Learn Dataset\\heart_disease_missing.csv')
heart_df_nona = heart_df.dropna()
features = ["exang","thal","slope","cp","oldpeak"]
heart_df_sub = heart_df_nona.copy()
for i in heart_df_nona.columns:
    if i not in features and i not in ["target"]:
        heart_df_sub.drop(columns = [i], inplace=True)

thal = [round(x) for x in list(heart_df_nona.loc[:,"thal"].values)]
heart_df_nona["thal"] = thal

outlier_ = iris_df_nona[iris_df_nona['petal_width']<0]
iris_df_nona.drop(index = list(outlier_.index), inplace=True)
```

CM6

Running model with default parameters

Iris dataset

Note that since data cleaning comes at the top of the ML pipeline, the data that is being taken forward has been cleaned by using the following methods,

1. Missing values --> dropped
2. Iris dataset : Outlier which is the negative entries in petal_width has been dropped
3. Heart dataset : Noise in thal has been normalized

In [3]:

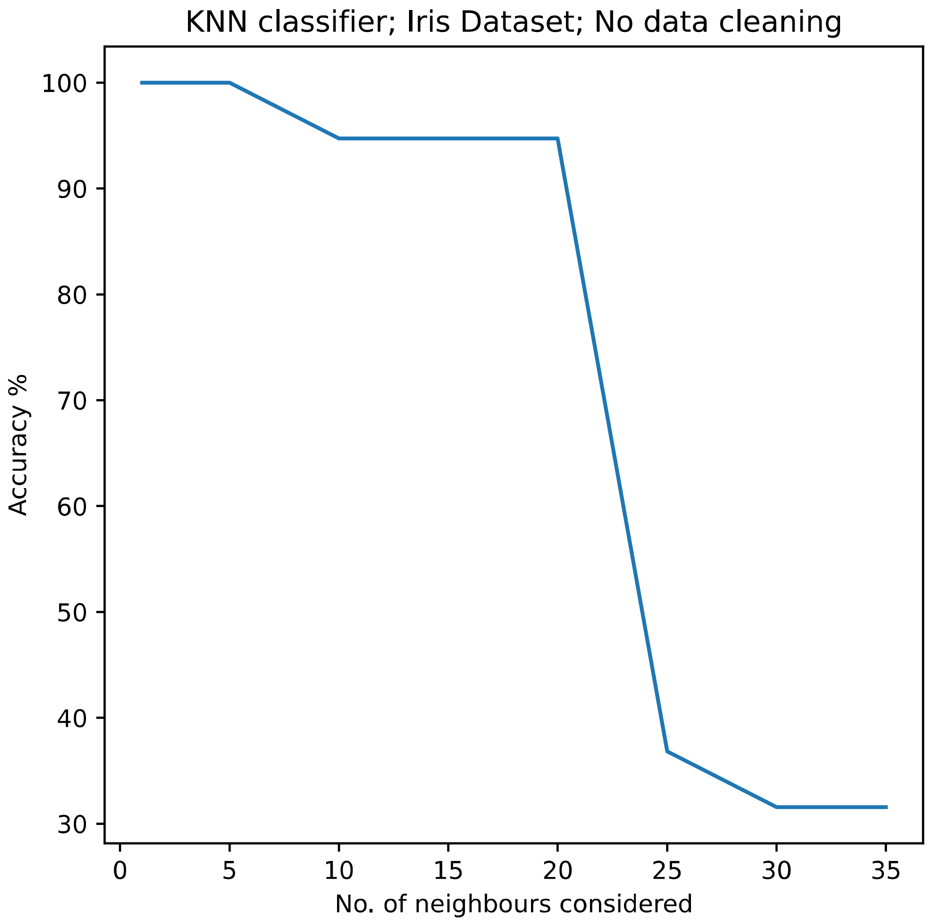
```
iris_df_nona_X = iris_df_nona.copy()
iris_df_nona_Y = iris_df_nona.copy()
iris_df_nona_X = iris_df_nona_X.drop(columns=["Class","species"])
iris_df_nona_Y = iris_df_nona_Y.drop(columns=["sepal_length","sepal_width","petal_length","petal_width","species"])

X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(iris_df_nona_X,iris_df_nona_Y, test_size = 0.4, random_state = 275)
X_test, X_val, Y_test, Y_val = sklearn.model_selection.train_test_split(X_test, Y_test, test_size = 0.5, random_state= 275)
k_n = [1, 5, 10, 15, 20, 25, 30, 35]
score_list = []
for k in k_n:
    knn_model = KNeighborsClassifier(n_neighbors= k)
    knn_model.fit(X_train,Y_train)
    Y_val_pred = knn_model.predict(X_val)
    pred_score = sklearn.metrics.accuracy_score(Y_val, Y_val_pred, normalize=True) * 100
    score_list.append(pred_score)

fig2 = plt.figure(figsize=(6,6))
plt.plot(k_n,score_list)
plt.xlabel('No. of neighbours considered')
plt.ylabel('Accuracy %')
plt.title('KNN classifier; Iris Dataset; No data cleaning')
```

Out[3]:

```
Text(0.5, 1.0, 'KNN classifier; Iris Dataset; No data cleaning')
```



In [4]:

```
best_k_index = score_list.index(max(score_list))
best_k = k_n[best_k_index]
print('Best K is...', best_k)
```

```
Best K is... 1
```

In [5]:

```
def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    #Does one hot encoding of the Y value
    lb = sklearn.preprocessing.LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)
    return sklearn.metrics.roc_auc_score(y_test, y_pred, average='macro')

knn_model = KNeighborsClassifier(n_neighbors=1)
knn_model.fit(X_train,Y_train)
Y_pred = knn_model.predict(X_test)
auc_score = multiclass_roc_auc_score(Y_test,Y_pred)
aucc_score = sklearn.metrics.accuracy_score(Y_test,Y_pred)
f_score = sklearn.metrics.f1_score(Y_test, Y_pred, average='macro')
print('AUC Score; K = 1; IRIS: ',auc_score)
print('Accuracy; K = 1, IRIS: ',aucc_score)
print('F-score; K = 1, IRIS: ', f_score)
```

```
AUC Score; K = 1; IRIS:  0.9121794871794872
Accuracy; K = 1, IRIS:  0.8888888888888888
F-score; K = 1, IRIS:  0.8777777777777779
```

Heart Disease Dataset

In [6]:

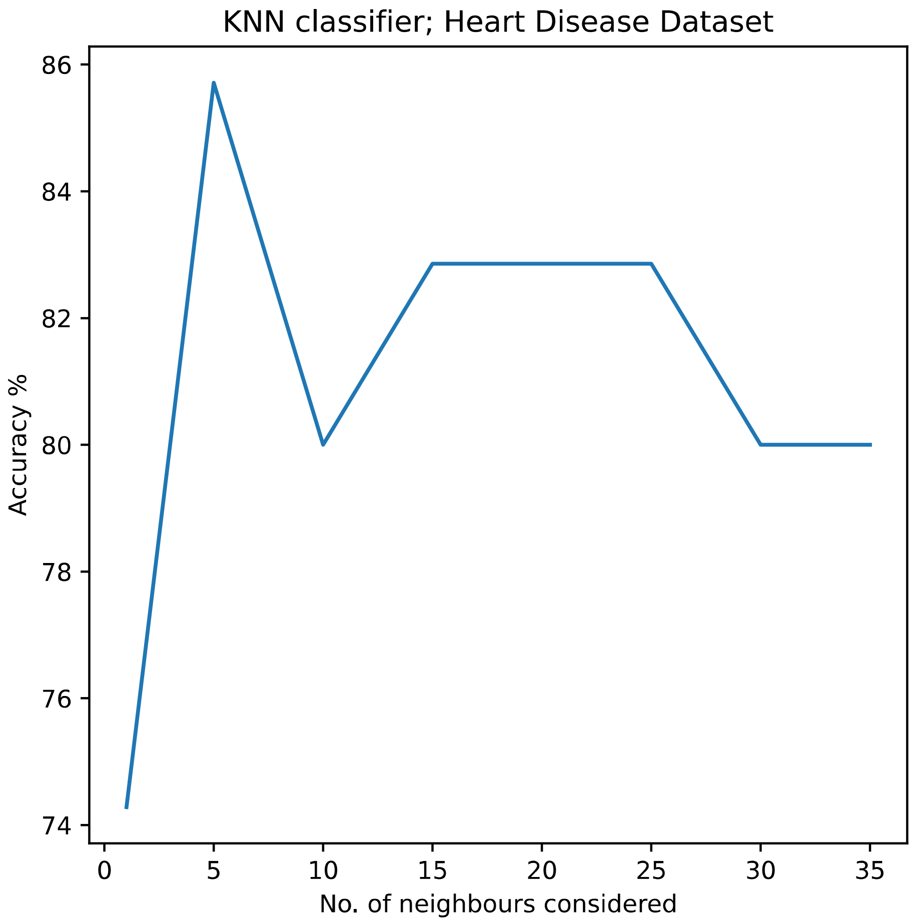
```
heart_df_X = heart_df_sub.copy().drop(columns=["target"])
heart_df_Y = heart_df_sub.copy().drop(columns=features)

X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_split(heart_df_X, heart_df_Y, test_size = 0.4, random_state = 275)
X_val, X_test, Y_val, Y_test = sklearn.model_selection.train_test_split(X_test, Y_test, test_size = 0.5, random_state= 275)

k_n = [1, 5, 10, 15, 20, 25, 30, 35]
score_list = []
for k in k_n:
    knn_model = KNeighborsClassifier(n_neighbors= k)
    knn_model.fit(X_train,Y_train)
    Y_val_pred = knn_model.predict(X_val)
    pred_score = sklearn.metrics.accuracy_score(Y_val, Y_val_pred, normalize=True) * 100
    score_list.append(pred_score)
fig2 = plt.figure(figsize=(6,6))
plt.plot(k_n,score_list)
plt.xlabel('No. of neighbours considered')
plt.ylabel('Accuracy %')
plt.title('KNN classifier; Heart Disease Dataset')
```

Out[6]:

```
Text(0.5, 1.0, 'KNN classifier; Heart Disease Dataset')
```



In [9]:

```
best_k_index = score_list.index(max(score_list))
best_k = k_n[best_k_index]
print('Best K is...', best_k)
```

```
Best K is... 5
```

In [11]:

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train,Y_train)
Y_pred = knn_model.predict(X_test)
auc_score = sklearn.metrics.roc_auc_score(Y_test, Y_pred, average='weighted')
aucc_score = sklearn.metrics.accuracy_score(Y_test,Y_pred)
f_score = sklearn.metrics.f1_score(Y_test, Y_pred, average='weighted')
print('AUC Score; K = 5; heart dataset...',auc_score)
print('Accuracy; K = 5, heart - dataset...',aucc_score)
print('F-score; K = 5, heart - dataset...', f_score)
```

```
AUC Score; K = 5; heart dataset... 0.7549342105263157
Accuracy; K = 5, heart - dataset... 0.7714285714285715
F-score; K = 5, heart - dataset... 0.7606393606393606
```

In [ ]: