In [3]:
```
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 [4]:
```
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"]
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)
```

CM5

Dealing with missing values

Iris dataset

In [5]:
```
iris_df.describe()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width
count	105.000000	101.000000	97.000000	105.000000
mean	5.858909	3.059083	3.812370	1.199708
std	0.861638	0.455116	1.793489	0.787193
min	4.344007	1.946010	1.033031	-0.072203
25%	5.159145	2.768688	1.545136	0.333494
50%	5.736104	3.049459	4.276817	1.331797
75%	6.435413	3.290318	5.094427	1.817211
max	7.795561	4.409565	6.768611	2.603123

We can see that the given iris dataset has missing values. We are going to remove any nan values to start with, which is an easy and straight forward solution. We will test our model with this method and choose whether to go for another method.

In [6]:
```
iris_df_nona = iris_df.dropna()
iris_df_nona.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	93.000000	93.000000	93.000000	93.000000
mean	5.867894	3.054935	3.808118	1.209826
std	0.892271	0.439463	1.811399	0.793656
min	4.344007	1.946010	1.033031	-0.072203
25%	5.152435	2.794790	1.541564	0.333494
50%	5.636744	3.049459	4.192791	1.369266
75%	6.478961	3.239682	5.098860	1.837925
max	7.795561	4.249211	6.768611	2.603123

Heart Disease Dataset

In [7]:
```
heart_df.describe()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	212.000000	212.000000	212.000000	205.000000	202.000000	212.000000	207.000000	208.000000	212.000000	200.000000	210.000000	212.000000	211.000000	212.000000
mean	54.311321	0.688679	0.957547	131.784610	244.133256	0.132075	0.560386	149.647978	0.344340	1.113106	1.423810	0.731132	2.349112	0.542453
std	9.145339	0.464130	1.022537	18.057222	46.444257	0.339374	0.535149	22.076206	0.476277	1.255908	0.623622	1.038762	0.602117	0.499374
min	29.000000	0.000000	0.000000	93.944184	126.085811	0.000000	0.000000	88.032613	0.000000	-0.185668	0.000000	0.000000	0.858554	0.000000
25%	47.000000	0.000000	0.000000	119.968114	211.969594	0.000000	0.000000	135.946808	0.000000	0.050778	1.000000	0.000000	1.949795	0.000000
50%	55.000000	1.000000	1.000000	130.010256	241.467023	0.000000	1.000000	151.939216	0.000000	0.726060	1.000000	0.000000	2.078759	1.000000
75%	61.000000	1.000000	2.000000	139.965470	272.484222	0.000000	1.000000	165.260092	1.000000	1.816733	2.000000	1.000000	2.970842	1.000000
max	77.000000	1.000000	3.000000	192.020200	406.932689	1.000000	2.000000	202.138041	1.000000	6.157114	2.000000	4.000000	3.277466	1.000000

Heart disease dataset also has quite a bit of missing values. We shall apply a similar approach to this dataset as well, which is to remove the nan entries.

In [8]:
```
heart_df_nona = heart_df.dropna()
heart_df_nona.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000
mean	54.649425	0.695402	0.948276	132.852160	246.142975	0.137931	0.563218	149.446186	0.367816	1.096423	1.396552	0.747126	2.351880	0.528736
std	9.275368	0.461565	1.015870	18.438982	46.541790	0.345823	0.541912	22.059644	0.483603	1.279873	0.634454	1.077552	0.613963	0.500614
min	29.000000	0.000000	0.000000	93.944184	126.085811	0.000000	0.000000	88.032613	0.000000	-0.176438	0.000000	0.000000	0.858554	0.000000
25%	47.250000	0.000000	0.000000	120.026582	213.267299	0.000000	0.000000	134.528882	0.000000	0.056498	1.000000	0.000000	1.963829	0.000000
50%	55.000000	1.000000	1.000000	130.021392	242.960083	0.000000	1.000000	151.490680	0.000000	0.676122	1.000000	0.000000	2.077903	1.000000
75%	61.750000	1.000000	2.000000	140.097261	273.983074	0.000000	1.000000	165.713572	1.000000	1.744327	2.000000	1.000000	2.970481	1.000000
max	77.000000	1.000000	3.000000	192.020200	406.932689	1.000000	2.000000	202.138041	1.000000	6.157114	2.000000	4.000000	3.277466	1.000000

Doing so, we have lost about 15% of our data.

Justification

Removing nan values is a straight forward method that we can adopt if we have few nan values. Numerical missing values are harder to fill with prediction and might at times prove to be too risky to fill it with mean without giving enough thought to it. Hence, when there are small number of missing values, it is simpler to just discard them. This is why we have adopted them. We have also dropped them in heart disease dataset.

Dealing with Outliers

Iris dataset

In [9]:
```
iris_df_nona.describe()
```

Out[9]:

	sepal_length	sepal_width	petal_length	petal_width
count	93.000000	93.000000	93.000000	93.000000
mean	5.867894	3.054935	3.808118	1.209826
std	0.892271	0.439463	1.811399	0.793656
min	4.344007	1.946010	1.033031	-0.072203
25%	5.152435	2.794790	1.541564	0.333494
50%	5.636744	3.049459	4.192791	1.369266
75%	6.478961	3.239682	5.098860	1.837925
max	7.795561	4.249211	6.768611	2.603123

The negative entry in petal_width is an outlier and can be removed.

In [10]:
```
outlier_ = iris_df_nona[iris_df_nona['petal_width']<0]
iris_df_nona.drop(index = list(outlier_.index), inplace=True)
iris_df_nona.describe()
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width
count	91.000000	91.000000	91.000000	91.000000
mean	5.890573	3.054944	3.859047	1.237675
std	0.888618	0.444313	1.797814	0.779370
min	4.344007	1.946010	1.033031	0.020731
25%	5.183002	2.783261	1.552927	0.341349
50%	5.695405	3.049459	4.276817	1.400355
75%	6.486531	3.265000	5.141844	1.860809
max	7.795561	4.249211	6.768611	2.603123

Heart Disease Dataset

In [11]:
```
heart_df_nona.describe()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000
mean	54.649425	0.695402	0.948276	132.852160	246.142975	0.137931	0.563218	149.446186	0.367816	1.096423	1.396552	0.747126	2.351880	0.528736
std	9.275368	0.461565	1.015870	18.438982	46.541790	0.345823	0.541912	22.059644	0.483603	1.279873	0.634454	1.077552	0.613963	0.500614
min	29.000000	0.000000	0.000000	93.944184	126.085811	0.000000	0.000000	88.032613	0.000000	-0.176438	0.000000	0.000000	0.858554	0.000000

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
25%	47.250000	0.000000	0.000000	120.026582	213.267299	0.000000	0.000000	134.528882	0.000000	0.056498	1.000000	0.000000	1.963829	0.000000
50%	55.000000	1.000000	1.000000	130.021392	242.960083	0.000000	1.000000	151.490680	0.000000	0.676122	1.000000	0.000000	2.077903	1.000000
75%	61.750000	1.000000	2.000000	140.097261	273.983074	0.000000	1.000000	165.713572	1.000000	1.744327	2.000000	1.000000	2.970481	1.000000
max	77.000000	1.000000	3.000000	192.020200	406.932689	1.000000	2.000000	202.138041	1.000000	6.157114	2.000000	4.000000	3.277466	1.000000

One outlier that we notice here is that thal, which is supposed to be categorical seems to be corrupted by noise which needs to be corrected. We will correct the same by rounding it off to the nearest whole number.

In [12]:
```
thal = [round(x) for x in list(heart_df_nona.loc[:,"thal"].values)]
heart_df_nona["thal"] = thal
heart_df_nona.describe()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000
mean	54.649425	0.695402	0.948276	132.852160	246.142975	0.137931	0.563218	149.446186	0.367816	1.096423	1.396552	0.747126	2.356322	0.528736
std	9.275368	0.461565	1.015870	18.438982	46.541790	0.345823	0.541912	22.059644	0.483603	1.279873	0.634454	1.077552	0.598206	0.500614
min	29.000000	0.000000	0.000000	93.944184	126.085811	0.000000	0.000000	88.032613	0.000000	-0.176438	0.000000	0.000000	1.000000	0.000000
25%	47.250000	0.000000	0.000000	120.026582	213.267299	0.000000	0.000000	134.528882	0.000000	0.056498	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.021392	242.960083	0.000000	1.000000	151.490680	0.000000	0.676122	1.000000	0.000000	2.000000	1.000000
75%	61.750000	1.000000	2.000000	140.097261	273.983074	0.000000	1.000000	165.713572	1.000000	1.744327	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	192.020200	406.932689	1.000000	2.000000	202.138041	1.000000	6.157114	2.000000	4.000000	3.000000	1.000000

Justification

In iris dataset, the count of outlier is very small and hence we are discarding them. In case of of thal, the entire dataset seems to be corrupted by noise and hence we are smoothening it by rounding it off to the nearest whole number.