Knearest Neighbors

In [18]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing, neighbors
import pandas as pd
```

In [8]:

```
df = pd.read_csv("../../Data-Sets/datasets/cancer/breast-cancer-wisconsin.data.txt", header=
df.head()
```

Out[8]:

	0	1	2	3	4	5	6	7	8	9	10
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

In [10]:

```
df.describe()
```

Out[10]:

	0	1	2	3	4	5	7	8	
count	6.990000e+02	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.
mean	1.071704e+06	4.417740	3.134478	3.207439	2.806867	3.216023	3.437768	2.866953	1.
std	6.170957e+05	2.815741	3.051459	2.971913	2.855379	2.214300	2.438364	3.053634	1.
min	6.163400e+04	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.
25%	8.706885e+05	2.000000	1.000000	1.000000	1.000000	2.000000	2.000000	1.000000	1.
50%	1.171710e+06	4.000000	1.000000	1.000000	1.000000	2.000000	3.000000	1.000000	1.
75%	1.238298e+06	6.000000	5.000000	5.000000	4.000000	4.000000	5.000000	4.000000	1.
max	1.345435e+07	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.

In [35]:

```
df.columns = ["name", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "class"]
```

In [36]:

```
df.head()
```

Out[36]:

	name	V1	V2	V3	V4	V5	V6	V7	V8	V9	class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

In [37]:

```
df = df.drop(["name"],1)
```

```
C:\Users\Kevin\AppData\Local\Temp/ipykernel_8724/1694563752.py:1: FutureWarning: In a future
version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be ke
yword-only
  df = df.drop(["name"],1)
```

In [38]:

```
df.replace("?", -99999, inplace=True) ## Reemplazamos todos los valores ?
```

In [39]:

```
Y = df["class"]
X = df[["V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9"]]
```

In [40]:

```
X.head()
```

Out[40]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	5	1	1	1	2	1	3	1	1
1	5	4	4	5	7	10	3	2	1
2	3	1	1	1	2	2	3	1	1
3	6	8	8	1	3	4	3	7	1
4	4	1	1	3	2	1	3	1	1

In [41]:

```
Y.head()
```

Out[41]:

```
0    2
1    2
2    2
3    2
4    2
Name: class, dtype: int64
```

Cuando el valor de la clase es 2, quere decir que es veningno. Cuando el tumor es 4 quiere decir que es maligno.

Clasificador de los K vecinos

In [42]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [43]:

```
clf = neighbors.KNeighborsClassifier()
```

In [71]:

```
clf.fit(X_train, Y_train)
```

Out[71]:

```
KNeighborsClassifier()
```

In [72]:

```
accuracy = clf.score(X_test, Y_test)
accuracy
```

Out[72]:

```
0.9571428571428572
```

Clasificación sin Limpieza

In [30]:

```
df = pd.read_csv("../../Data-Sets/datasets/cancer/breast-cancer-wisconsin.data.txt", header=N
df.replace("?", -99999, inplace=True) ## Reemplazamos todos los valores ?
df.columns = ["name", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "class"]
Y = df["class"]
X = df[["name","V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9"]]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
clf = neighbors.KNeighborsClassifier()
clf.fit(X_test, Y_test)
accuracy = clf.score(X_test, Y_test)
accuracy
```

Out[30]:

```
0.7357142857142858
```

Fijarse cómo la eficiencia del Kneighbors funciona con un **99,28%** de eficacia cuando filtramos los predictores y cómo al ingresarle los nombres cae a un **73%**. Por ende no tenemos que suministrar absolutamente todos los datos de entrada cuando hacemos una clasificación con K vecinos.

Clasificación de nuevos datos

In [47]:

```
sample_measure = np.array([4,2,1,1,1,2,3,2,1]).reshape(1,-1)
predict = clf.predict(sample_measure)
```

In [48]:

```
predict
```

Out[48]:

```
array([2], dtype=int64)
```

Con estos datos tenemos la seguridad que la célula es veningna, o sea buena.

In [67]:

```
sample_measure2 = np.array([[4,2,1,1,1,2,3,2,1], [4,2,1,1,1,2,3,2,1]]).reshape(2,-1)
```

In [68]:

```
predict = clf.predict(sample_measure2)
```

In [69]:

```
predict
```

Out[69]:

```
array([2, 2], dtype=int64)
```

Ambos elementos entran en la misma clase.

In [ ]: