Ejercicio 1

1.- Efectuad, si es necesario, el tratamiento previo de los datos. Justificad todas las decisiones que toméis.

El archivo proporcionado no tiene valores ausentes, por tanto no es necesario hacer un tratamiento en este sentido. No todos los datos están expresados numéricamente, la categoría SAFETY la hemos convertido a valores numéricos equidistantes, de manera que no afecten al conjunto de datos cuando se estandaricen. El criterio escogido es 0 para low, 1 para med y 2 para high. También se ha convertido la categoría CLASS con un criterio binario para acc o unacc. Se han estandarizado los datos con un centro en 0, de manera que valores absolutos muy altos como PRICE no afecten al resultado.

Las medias y las desviaciones típicas son:
theMeans: [2.70 1.35 3.10 3.40 1.50]
theStd: [1.10 5.93 1.13 1.20 6.70]

Los datos estandarizados son los siguientes:
[[1.18 1.09 -0.97 -1.17 -2.24]
[0.27 -1.43 -0.09 1.33 0.75]
[-1.55 1.09 1.67 0.50 0.75]
[-0.64 -0.59 -0.97 0.50 -0.75]
[1.18 1.09 1.67 -1.17 0.75]
[0.27 -0.59 0.79 -1.17 0.75]
[0.27 1.09 -0.09 0.50 0.75]
[-0.64 0.25 -0.97 -1.17 -0.75]
[-1.55 -1.43 -0.09 1.33 -0.75]
[1.18 -0.59 -0.97 0.50 0.75]]

2.- Utilizad el k-means nítido para categorizar los datos del archivo mencionado en dos categorías, ignorando las columnas no pertinentes. ¿Cuál es el nivel de precisión[1] del resultado?

Aplicamos k-means nítido utilizando la distancia euclidea, y k=2, dado que se buscan dos categorías. Para los primeros centroides se utilizan las dos primeras filas del conjunto de datos. He observado que si las filas son aleatorias los centroides varían. Marcamos una tolerancia muy baja para asegurarnos de que llegamos al mismo resultado que el obtenido usando scikit. No obstante se introduce un número máximo de iteraciones (al cual nunca llegamos). Los pasos se pueden observar en el algoritmo desarrollado, pero brévemente son los siguientes:

- Para el número máximo de iteraciones, mientras no se llegue la optimización:
  - Se inicializan las clasificaciones.
  - Para el número k veces:
    - Se calculan la distancias de cada fila de datos sobre sus propios centroides.
    - Sacamos el índice de la distancia mínima, que corresponderá al grupo de pertenencia de esa fila.
    - Guardamos la relación de esa fila a ese grupo de pertenencia.
  - Calculamos los nuevos centroides en base a las nuevas filas de pertenencia que hemos hayado antes mediante una media de las observaciones.
  - Actualizamos los centroides.
  - Si los centroides no han cambiado, hemos alcanzado la optimización y finalizamos.

Para la obtención del resultado, he reutilizado el formato que se provee en kmeans-sklearn.py, de manera que sea posible comprobar que la implementación es correcta, a la vez que resulta fácil mostrar los resultados obtenidos.

El resultado obtenido es el siguiente:

```
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS        : [1, 5, 8]
 - CLASS 0 CENTROID       : [0.58 0.81 -0.09 -1.17 -0.75]
 - CLASS 1 MEMBERS        : [2, 3, 4, 6, 7, 9, 10]
 - CLASS 1 CENTROID       : [-0.25 -0.35 0.04 0.50 0.32]
[K-MEANS PERFORMANCE]
 - ACCURACY               : 80.0%
```

Utilizando todos los datos alcanzamos una precisión del 80% ,y la distribución de los ejemplos más los centroides tal y como se muestra en los datos de arriba.

Tal y como indicaba al principio, si los centroides iniciales se escogen aleatoriamente entre el conjunto de datos, los centroides varían, y también la pertenencia de los ejemplos, aunque en este caso no la precisión:

```
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS        : [1, 4, 5, 6, 8]
 - CLASS 0 CENTROID       : [0.27 0.25 -0.09 -0.83 -0.45]
 - CLASS 1 MEMBERS        : [2, 3, 7, 9, 10]
 - CLASS 1 CENTROID       : [-0.27 -0.25 0.09 0.83 0.45]
[K-MEANS PERFORMANCE]
 - ACCURACY               : 80.0%
```

3.- Aplicad el algoritmo PCA para reducir la dimensionalidad del conjunto anterior conservando el 95% de la varianza. Utilizad k-means nítido sobre el conjunto reducido de la misma forma que en el apartado anterior. Comparad los resultados.

Aplicando PCA, obtenemos los siguientes datos intermedios:

```
theMeans [27000.00 1350.00 3.10 3.40 1.50]
theStd [11000.00 593.72 1.14 1.20 0.67]

Standarized data:
[[1.18 1.09 -0.97 -1.17 -2.24]
 [0.27 -1.43 -0.09 1.33 0.75]
 [-1.55 1.09 1.67 0.50 0.75]
 [-0.64 -0.59 -0.97 0.50 -0.75]
 [1.18 1.09 1.67 -1.17 0.75]
 [0.27 -0.59 0.79 -1.17 0.75]
 [0.27 1.09 -0.09 0.50 0.75]
 [-0.64 0.25 -0.97 -1.17 -0.75]
 [-1.55 -1.43 -0.09 1.33 -0.75]
 [1.18 -0.59 -0.97 0.50 0.75]]

Covariance:
[[1.00 0.24 -0.14 -0.44 0.07]
 [0.24 1.00 0.32 -0.55 -0.06]
 [-0.14 0.32 1.00 -0.10 0.59]
 [-0.44 -0.55 -0.10 1.00 0.25]
 [0.07 -0.06 0.59 0.25 1.00]]

Eigen Values:
[1.87 1.63 0.91 0.40 0.19]
```

```
Eigen Vectors:
[[-0.45 0.13 0.78 -0.14 0.39]
 [-0.59 -0.12 -0.37 -0.69 -0.14]
 [-0.17 -0.70 -0.23 0.27 0.59]
 [0.64 -0.13 0.09 -0.66 0.37]
 [0.07 -0.68 0.43 -0.05 -0.59]]

Explained Variances:
[0.37 0.70 0.88 0.96 1.00]
[[-0.45 0.13 0.78]
 [-0.59 -0.12 -0.37]
 [-0.17 -0.70 -0.23]
 [0.64 -0.13 0.09]
 [0.07 -0.68 0.43]]

[PCA OUTPUT]
 - CUMULATIVE VARIANCES: [0.37 0.70 0.88 0.96 1.00]
 - VECTORS REQUIRED    : 4
 - PROJECTED ATTRIBUTES:
[[-1.93 2.37 -0.33 -0.30]
 [1.65 -0.42 1.21 0.01]
 [0.14 -2.07 -1.64 -0.45]
 [1.07 1.11 -0.33 -0.06]
 [-2.16 -1.51 0.34 0.26]
 [-0.60 -0.80 0.46 1.31]
 [-0.38 -0.61 0.19 -1.18]
 [-0.50 1.23 -0.80 0.45]
 [2.36 0.37 -0.85 0.34]
 [0.36 0.32 1.74 -0.39]]
```

Es decir, que con las cuatro primeras columnas vamos a mantener el 95% de varianza. Si utilizamos los nuevos atributos en k-means el resultado que veremos tendrá la misma eficiencia pero utilizando menos datos. Aquí se puede observar una comparativa de datos:

Este sería el resultado utilizando todo el conjunto de datos:

```
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS       : [1, 5, 8]
 - CLASS 0 CENTROID      : [0.58 0.81 -0.09 -1.17 -0.75]
 - CLASS 1 MEMBERS       : [2, 3, 4, 6, 7, 9, 10]
 - CLASS 1 CENTROID      : [-0.25 -0.35 0.04 0.50 0.32]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 80.0%
```

Utilizando los atributos obtenidos con PCA tenemos unos nuevos centroides que son capaces de categorizar con la misma precisión que en el caso anterior.

```
[K-MEANS OUTPUT]
 - CLASS 0 MEMBERS       : [1, 5, 8]
 - CLASS 0 CENTROID      : [-1.53 0.70 -0.26 0.14]
 - CLASS 1 MEMBERS       : [2, 3, 4, 6, 7, 9, 10]
 - CLASS 1 CENTROID      : [0.66 -0.30 0.11 -0.06]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 80.0%
```

NOTA: el archivo K_Means.py desarrolla el algoritmo k-means y el archivo PCA.py desarrolla la obtención de los nuevos atributos.

Ejercicio 2

Las distancias obtenidas para el enlace promedio:

	1	2	3	4	5	6	7	8	9
0	4,81	5,11	3,34	3,98	3,96	3,64	2,50	4,80	3,81
1		3,67	2,29	4,07	2,78	2,66	3,59	2,35	1,73
2			3,59	3,20	3,11	2,53	3,68	3,52	4,15
3				4,26	2,99	2,58	1,87	1,73	2,35
4					2,11	2,59	3,64	5,04	3,55
5						2,53	2,62	3,64	2,59
6							2,70	3,55	2,11
7								3,27	3,00
8									3,44

{1,9}-average

	2	3	4	5	6	7	8	1-9
0	5,11	3,34	3,98	3,96	3,64	2,50	4,80	3,81
2		3,59	3,20	3,11	2,53	3,68	3,52	4,15
3			4,26	2,99	2,58	1,87	1,73	2,35
4				2,11	2,59	3,64	5,04	3,55
5					2,53	2,62	3,64	2,59
6						2,70	3,55	2,11
7							3,27	3,00
8								3,44

{3-8}-average

	2	4	5	6	7	3-8	1-9
0	5,11	3,98	3,96	3,64	2,50	3,34	3,81
2		3,20	3,11	2,53	3,68	3,52	4,15
4			2,11	2,59	3,64	5,04	3,55
5				2,53	2,62	3,64	2,59
6					2,70	3,55	2,11
7						3,27	3,00
3-8							2,35

{4-5}-average

	2	4-5	6	7	3-8	1-9
0	5,11	3,96	3,64	2,50	3,34	3,81
2		3,11	2,53	3,68	3,52	4,15
4-5			2,56	3,13	4,34	3,07
6				2,70	3,55	2,11
7					3,27	3,00
3-8						2,35

{1-9-6}-average

	2	4-5	7	3-8	1-6-9
0	5,11	3,96	2,50	3,34	3,725
2		3,11	3,68	3,52	3,34
4-5			3,13	4,34	2,815
7				3,27	3,00
3-8					2,35

{0-7}-average

	4-5	0-7	3-8	1-6-9
2	3,11	3,68	3,52	3,34
4-5		3,13	4,34	2,815
0-7			3,305	3,3625
3-8				2,35

{1-4-5-6-9}-average

	0-7	3-8	1-4-5-6-9
2	3,68	3,52	3,225
0-7		3,305	3,3625
3-8			2,35

{1-3-4-5-6-8-9}-average

	0-7	1-3-4-5-6-8-9
2	3,68	3,3725
0-7		3,3625

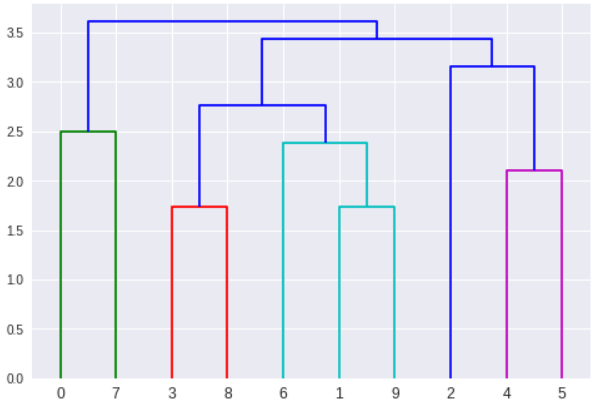
{0-7} {1-2-3-4-5-6-8-9}-average

	0-7	1-2-3-4-5-6-8-9
0-7		3,3625

Por tanto si consideramos dos categorías, las agrupaciones son {0-7} y {1-2-3-4-5–6-8-9}.
La precisión sería = (2+5)/10 = 70%, y el dendograma se muestra a continuación:

{0-7} naranja {1-2-4-5-6-8-9} verde

PRICE	MAINT	DOORS	PERSONS	SAFETY	CLASS
40000	2000	2	2	low	unacc
30000	500	3	5	high	acc
10000	2000	5	4	high	acc
20000	1000	2	4	med	acc
40000	2000	5	2	high	unacc
30000	1000	4	2	high	unacc
30000	2000	3	4	high	unacc
20000	1500	2	2	med	unacc
10000	500	3	5	med	acc
40000	1000	2	4	high	acc



Las distancias obtenidas para el enlace simple:

	1	2	3	4	5	6	7	8	9
0	4,81	5,11	3,34	3,98	3,96	3,64	2,50	4,80	3,81
1		3,67	2,29	4,07	2,78	2,66	3,59	2,35	1,73
2			3,59	3,20	3,11	2,53	3,68	3,52	4,15
3				4,26	2,99	2,58	1,87	1,73	2,35
4					2,11	2,59	3,64	5,04	3,55
5						2,53	2,62	3,64	2,59
6							2,70	3,55	2,11
7								3,27	3,00
8									3,44

{1,9}-simple

	2	3	4	5	6	7	8	1-9
0	5,11	3,34	3,98	3,96	3,64	2,50	4,80	3,81
2		3,59	3,20	3,11	2,53	3,68	3,52	4,15
3			4,26	2,99	2,58	1,87	1,73	2,35
4				2,11	2,59	3,64	5,04	3,55
5					2,53	2,62	3,64	2,59
6						2,70	3,55	2,11
7							3,27	3,00
8								3,44

{3-8}-simple

	2	4	5	6	7	3-8	1-9
0	5,11	3,98	3,96	3,64	2,50	3,34	3,81
2		3,20	3,11	2,53	3,68	3,52	4,15
4			2,11	2,59	3,64	5,04	3,55
5				2,53	2,62	3,64	2,59
6					2,70	3,55	2,11
7						3,27	3,00
3-8							2,35

{1-9-6}-simple

	2	4	5	7	3-8	1-9-6
0	5,11	3,98	3,96	2,50	3,34	3,64
2		3,20	3,11	3,68	3,52	2,53
4			2,11	3,64	5,04	2,59
5				2,62	3,64	2,53
7					3,27	3,00
3-8						2,35

{4-5}-simple

	2	4-5	7	3-8	1-6-9
0	5,11	3,96	2,50	3,34	3,64
2		3,11	3,68	3,52	2,53
4-5			2,62	3,64	2,53
7				3,27	3,00
3-8					2,35

{0-7}-simple

	4-5	0-7	3-8	1-6-9
2	3,11	3,68	3,52	2,53
4-5		2,62	3,64	2,53
0-7			3,27	3,00
3-8				2,35

{1-9-6-3-8}-simple

	4-5	0-7	1-9-6-3-8
2	3,11	3,68	2,53
4-5		2,62	2,53
0-7			3,00

{0-1-9-6-7-3-8}-simple

	0-7	0-1-9-6-3-8-4-5
2	3,68	2,53
0-7		3,00

{0-7} {1-2-3-4-5-6-8-9}-simple

	0-7	1-2-3-4-5-6-8-9
0-7		3,00

La agrupación resultante es la misma, por tanto con el mismo dendograma y misma precisión.

Para el completo:

	1	2	3	4	5	6	7	8	9
0	4,81	5,11	3,34	3,98	3,96	3,64	2,50	4,80	3,81
1		3,67	2,29	4,07	2,78	2,66	3,59	2,35	1,73
2			3,59	3,20	3,11	2,53	3,68	3,52	4,15
3				4,26	2,99	2,58	1,87	1,73	2,35
4					2,11	2,59	3,64	5,04	3,55
5						2,53	2,62	3,64	2,59
6							2,70	3,55	2,11
7								3,27	3,00
8									3,44

{1,9}-completo

	2	3	4	5	6	7	8	1-9
0	5,11	3,34	3,98	3,96	3,64	2,50	4,80	4,81
2		3,59	3,20	3,11	2,53	3,68	3,52	4,15
3			4,26	2,99	2,58	1,87	1,73	2,35
4				2,11	2,59	3,64	5,04	3,55
5					2,53	2,62	3,64	2,59
6						2,70	3,55	2,11
7							3,27	3,00
8								3,44

{3-8}-completo

	2	4	5	6	7	3-8	1-9
0	5,11	3,98	3,96	3,64	2,50	4,80	4,81
2		3,20	3,11	2,53	3,68	3,59	4,15
4			2,11	2,59	3,64	5,04	3,55
5				2,53	2,62	3,64	2,59
6					2,70	3,55	2,11
7						3,27	3,00
3-8							3,44

{1-9-6}-completo

	2	4	5	7	3-8	1-9-6
0	5,11	3,98	3,96	2,50	4,80	4,81
2		3,20	3,11	3,68	3,59	4,15
4			2,11	3,64	5,04	3,55
5				2,62	3,64	2,53
7					3,27	3,00
3-8						3,44

{4-5}-completo

	2	4-5	7	3-8	1-6-9
0	5,11	3,98	2,50	4,80	4,81
2		3,20	3,68	3,59	4,15
4-5			3,64	5,04	3,55
7				3,27	3,00
3-8					3,44

{0-7}-completo

	4-5	0-7	3-8	1-6-9
2	3,20	3,68	3,59	4,15
4-5		3,64	5,04	3,55
0-7			3,27	3,00
3-8				3,44

{1-9-6-3-8}-completo

	4-5	3-8	1-9-6-0-7
2	3,20	3,59	4,15
4-5		5,04	3,64
3-8			3,27

{0-1-9-6-7-3-8}-completo

	4-5	1-9-6-0-7-3-8
2	3,20	4,15
4-5		5,04

{4-5-2} {1-9-6-0-7-3-8}-completo

	4-5-2	1-9-6-0-7-3-8
4-5-2	3,20	5,04

La precisión es de (4+2)/10 = 60%

{4-5-2} naranja {1-9-6-0-7-3-8} verde

PRICE	MAINT	DOORS	PERSONS	SAFETY	CLASS
40000	2000	2	2	low	unacc
30000	500	3	5	high	acc
10000	2000	5	4	high	acc
20000	1000	2	4	med	acc
40000	2000	5	2	high	unacc
30000	1000	4	2	high	unacc
30000	2000	3	4	high	unacc
20000	1500	2	2	med	unacc
10000	500	3	5	med	acc
40000	1000	2	4	high	acc

Ejercicio 3

Tal y como se indica en el enunciado se escogen diferentes métodos, que en este caso son k-means++, random y MiniBatchKMeans. Todas las líneas de código correspondientes se muestran en el archivo python Cars.py, no obstante los resultados y la invocación de los algoritmos son los siguientes:

1.- K-Means:
kMeansOut=KMeans(n_clusters=2,init=sampleData[0:2,:],max_iter=25).fit(sampleData)

```
[K-MEANS OUTPUT]
 - CLASS 0 CENTROID      : [-0.00 -0.00 0.00 0.00 -1.22]
 - CLASS 1 CENTROID      : [-0.00 -0.00 -0.00 0.00 0.61]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 63.31018518518518%
 - CONFUSION MATRIX      : [[576.00 634.00] [0.00 518.00]]
 - FAILS                 : 634
```

2.- K-Means with PCA attributes:
kMeansOut=KMeans(n_clusters=2,init=sampleData[0:2,:],max_iter=25).fit(sampleData)
(En este caso lo único que cambia respecto del anterior son los datos sumisistrados)

```
[K-MEANS OUTPUT]
 - CLASS 0 CENTROID      : [0.00 -0.00 -0.00 0.00 -1.22]
 - CLASS 1 CENTROID      : [-0.00 0.00 0.00 -0.00 0.61]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 63.31018518518518%
 - CONFUSION MATRIX      : [[576.00 634.00] [0.00 518.00]]
 - FAILS                 : 634
```

3.- K-Means++:
kMeansOut=KMeans(n_clusters=2,init='k-means++',max_iter=25).fit(sampleData)

```
[K-MEANS OUTPUT]
 - CLASS 0 CENTROID      : [-0.00 -0.89 -0.00 -0.00 0.00]
 - CLASS 1 CENTROID      : [-0.00 0.89 -0.00 -0.00 0.00]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 42.01388888888889%
 - CONFUSION MATRIX      : [[536.00 674.00] [328.00 190.00]]
 - FAILS                 : 1002
```

4.- K-Means random:
kMeansOut=KMeans(n_clusters=2,init='random',max_iter=25).fit(sampleData)

```
[K-MEANS OUTPUT]
 - CLASS 0 CENTROID      : [-0.00 -0.00 0.00 -1.34 0.00]
 - CLASS 1 CENTROID      : [-0.00 -0.00 -0.00 0.67 0.00]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 63.31018518518518%
 - CONFUSION MATRIX      : [[576.00 634.00] [0.00 518.00]]
 - FAILS                 : 634
```

5.- Mini Batch K-Means:

```
from random import randint
(…)
random_state = np.random.RandomState(0)
miniBatchKMeans = MiniBatchKMeans(n_clusters=2, init='random', n_init=1,
random_state=random_state).fit(sampleData)

[K-MEANS OUTPUT]
 - CLASS 0 CENTROID      : [0.02 0.88 -0.35 -0.13 -0.39]
 - CLASS 1 CENTROID      : [-0.01 -0.57 0.21 0.11 0.24]
[K-MEANS PERFORMANCE]
 - ACCURACY              : 62.326388888888886%
 - CONFUSION MATRIX      : [[665.00 545.00] [106.00 412.00]]
 - FAILS                 : 651
```

Ejercicio 4

La aplicación de PCA junto con KMeans es interesante porque reduce el número de características manteniendo la varianza, eliminando así información irrelevante y ruido. El problema observo es el coste calcular el nuevo conjunto reducido. Además, los datos han cambiado y ello podría traer dificultades de análisis. En cuanto a KMeans notar que tiene el inconveniente de que según el conjunto inicial, obtendremos un resultado diferente

Aun así veo que en la actualizad es una práctica muy habitual el utilizar PCA y KMeans conjuntamente. Por otra parte los dendogramas tienen una complejidad de O(2n), menor que KMeans O(t·k·n·d) (NP-HARD), obteniendo precisiones superiores o similares.