Alejandro Arias Zuluaga

201711999

Reto 6: Clasificación de Géneros Musicales (*on-line backpropagation*)

En esta ocasión, se realizó el mismo reto 5, correspondiente a clasificación binaria de dos grupos de géneros musicales (‘*soul and reggae’*=1 vs ‘*jazz and blues*’=0); sin embargo, se realizó una implementación del método de *on-line backpropagation* directamente en el lenguaje de programación *Python*, el planteamiento realizado fue basado en las diapositivas de clase [1] así como en el código analizado en la fuente [2]. A continuación, se describirá el proceso mediante el cual se desarrolló este método para, finalmente, obtener una red neuronal apropiada para el modelo.

En primer lugar, se definió la red neuronal como un arreglo de capas, cada una de las cuales contiene cierto número de neuronas (las capas escondidas poseen un tamaño dado por un parámetro que se desea modificar posteriormente), e igualmente, cada neurona corresponde a un “diccionario” con los siguientes parámetros:

‘weights’: pesos dados a cada uno de los parámetros recibidos por la neurona, estos son inicializados aleatoriamente en un rango entre 0 y 1.

‘*z’*: salida de la neurona luego de pasar por su correspondiente función de activación.

‘*delta’*: error propagado hasta la neurona multiplicado por la derivada de la función de activación evaluada en el valor de *‘z’* obtenido.

En segundo lugar, se desarrolló una función para la obtención de la salida de la red (*forward pass*) de forma progresiva a lo largo de cada capa escondida. Para esto, en cada neurona se definió la misma función de activación (en este caso, una sigmoide, debido a la simplicidad de su derivada $z * (1 - z)$), cuya entrada corresponde a una combinación lineal entre las componentes de la entrada y los pesos de la neurona (uno de estos pesos se asocia con un parámetro de la entrada constante con valor 1). Por lo tanto, es claro que todas las neuronas pertenecientes a la misma capa poseen las mismas entradas pero distintas salidas (todos sus pesos poseen valores distintos), las cuales, a su vez, equivalen a las entradas de la siguiente capa hasta llegar a la última capa de salida. Esta capa de salida únicamente se encuentra compuesta por 1 neurona en todos los casos, ya que el problema a resolver corresponde a clasificación binaria (la salida de la red corresponde a la probabilidad de que la entrada sea clasificada como 1).

En tercer lugar, el proceso de propagación del error se desarrolla necesariamente después de obtener esta salida de la red, a partir de la cual se calcula el error en la capa/neurona de salida, y luego calcular el valor de su correspondiente *‘delta’*. Acto seguido, el cálculo del error propagado hacia la capa anterior corresponde a la suma ponderada entre los *‘delta’* de las neuronas de la siguiente capa y cada uno de los pesos que se conectan a estas, luego, nuevamente se multiplica por la derivada de la sigmoide evaluada en la salida de cada neurona de la capa anterior. Este proceso se realiza de forma análoga para todas las capas y, una vez obtenidos y almacenados todos los *‘delta’* de las neuronas, se realiza una actualización de los pesos de cada una mediante la suma del factor $\eta * \delta * z$, empleando η como la tasa de aprendizaje, δ correspondiente al *‘delta’* de la neurona actual y z como la entrada recibida de la capa anterior, la cual fue multiplicada por el peso que se está actualizando.

Finalmente, este proceso se repite cierto número de iteraciones para todos los datos de entrenamiento seleccionados, realizando una evaluación de la precisión de la red en este mismo

conjunto de datos y su constante mejora, a medida que se actualizan los pesos de las neuronas. Este proceso se puede observar en las figuras 1, 3 y 5.

Adicionalmente, se evaluó la precisión de cada red neuronal, variando el tamaño de las capas escondidas para obtener las gráficas observadas en las figuras 2, 4 y 6.

Entrenamiento con 1 capa escondida:

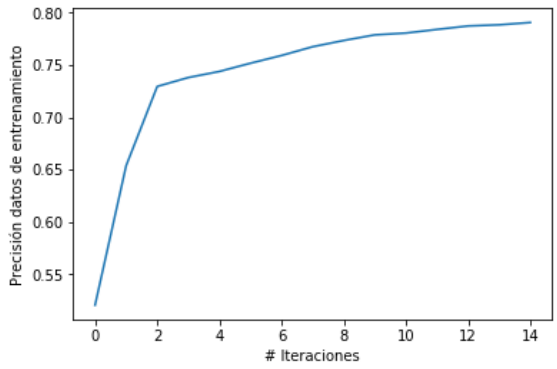


Figura 1. Entrenamiento 1 capa escondida.

Resultados empleando 1 capa escondida:

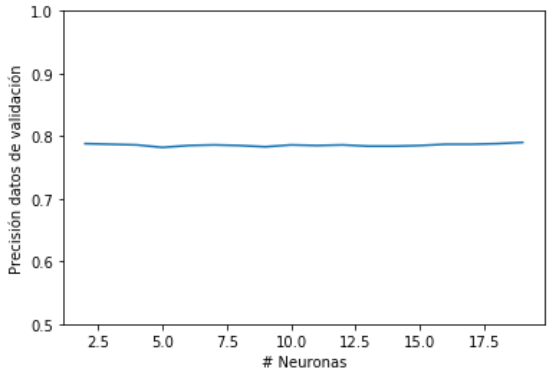


Figura 2. Resultados 1 capa escondida.

Entrenamiento con 2 capas escondidas:

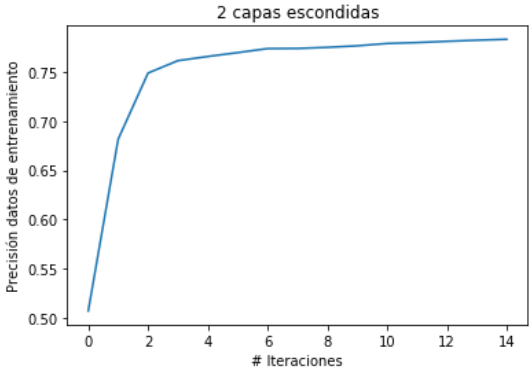


Figura 3. Entrenamiento 2 capas escondidas.

Resultados empleando 2 capas escondidas:

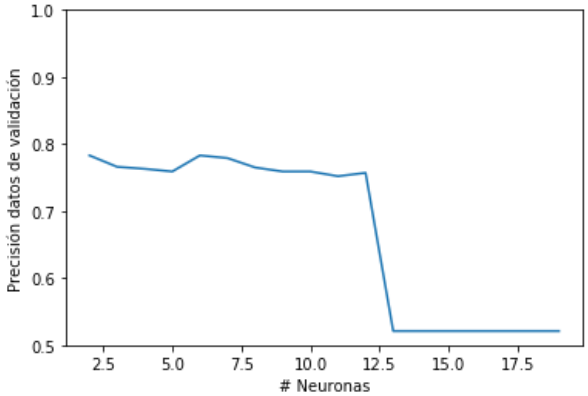


Figura 4. Resultados 2 capas escondidas.

Entrenamiento con 3 capas escondidas:

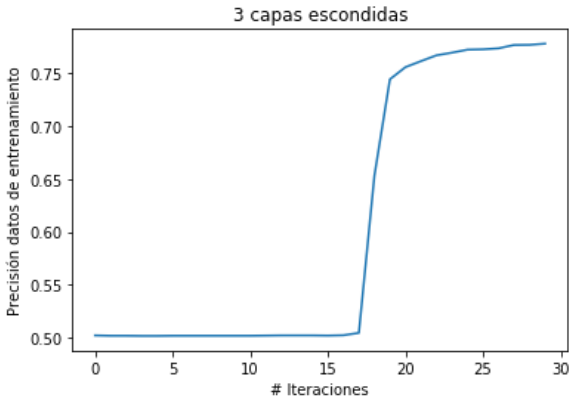


Figura 5. Entrenamiento 3 capas escondidas.

Resultados empleando 3 capas escondidas:

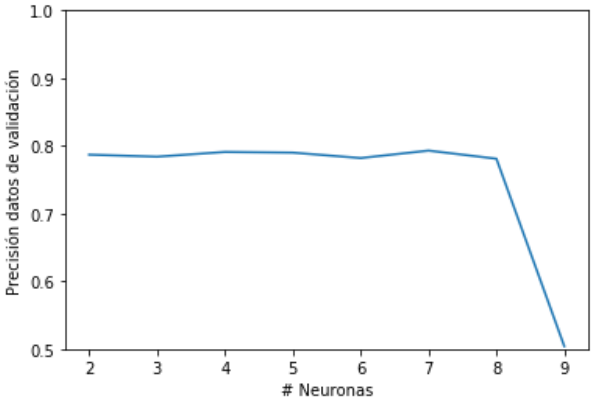


Figura 6. Resultados 3 capas escondidas.

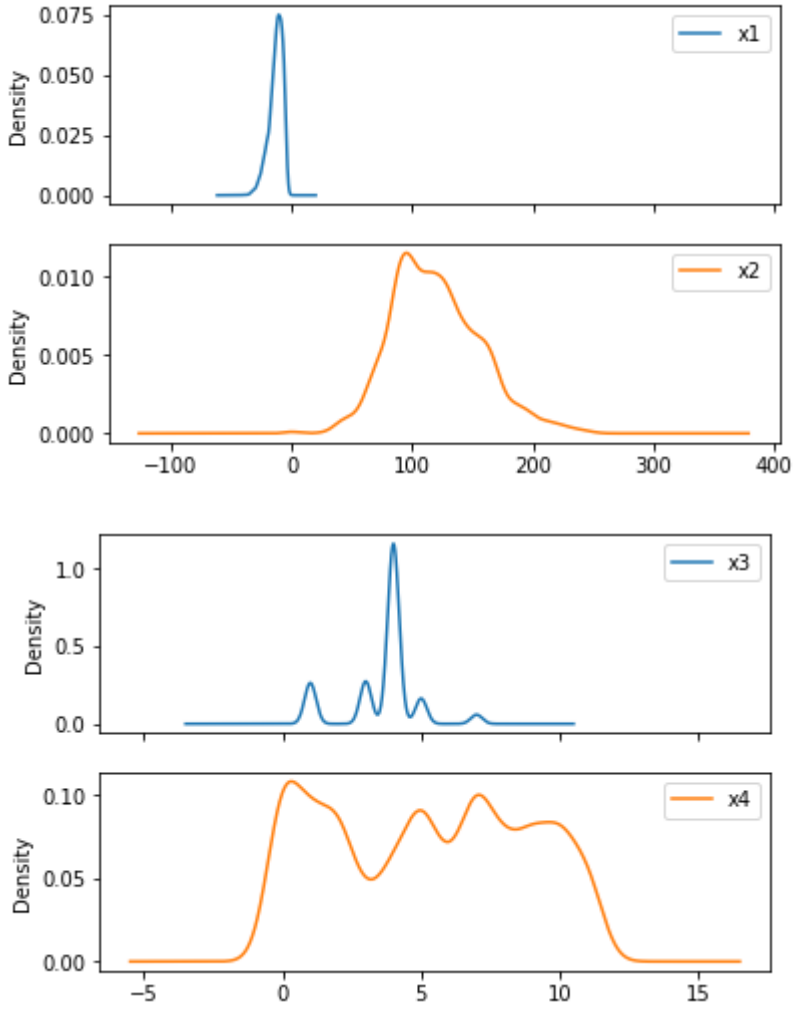
Referencias

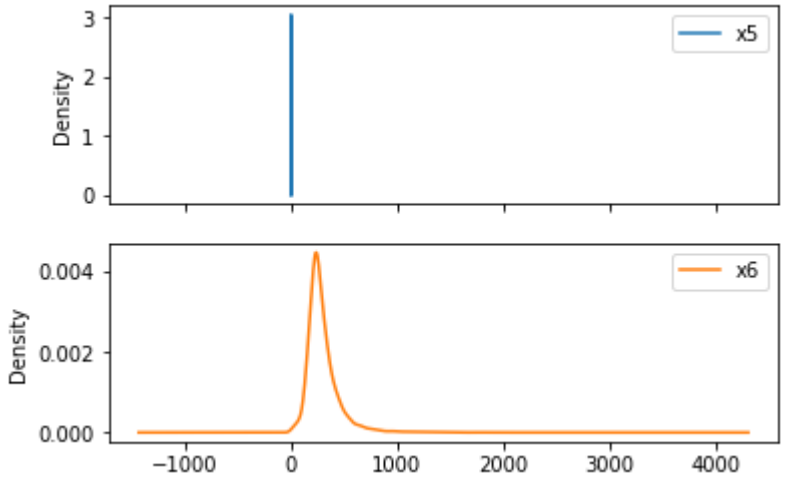
[1] Lozano, Fernando. Redes neuronales. Universidad de los Andes (2019)

[2] How to Code a Neural Network with Backpropagation In Python. Disponible en línea en: https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

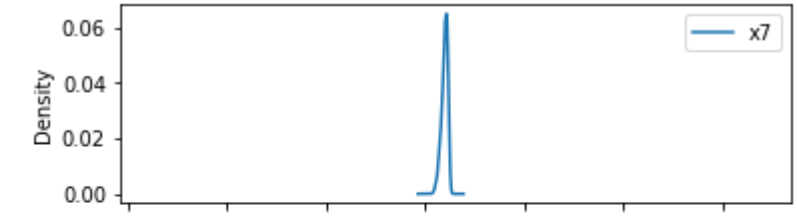
ANEXOS

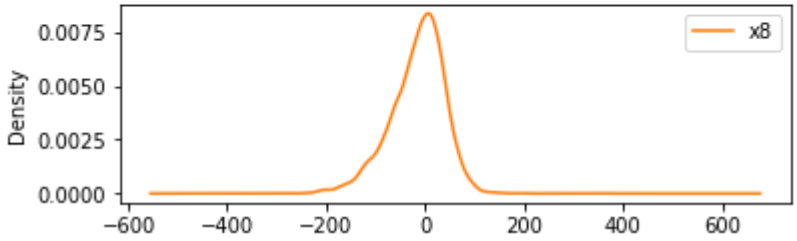
Gráficas de densidad de distribución de los parámetros descriptores

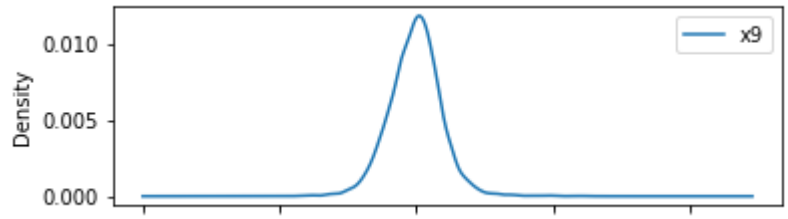


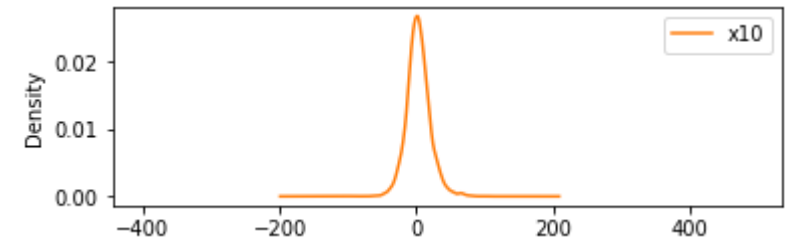

x5
Density
3
2
1
0

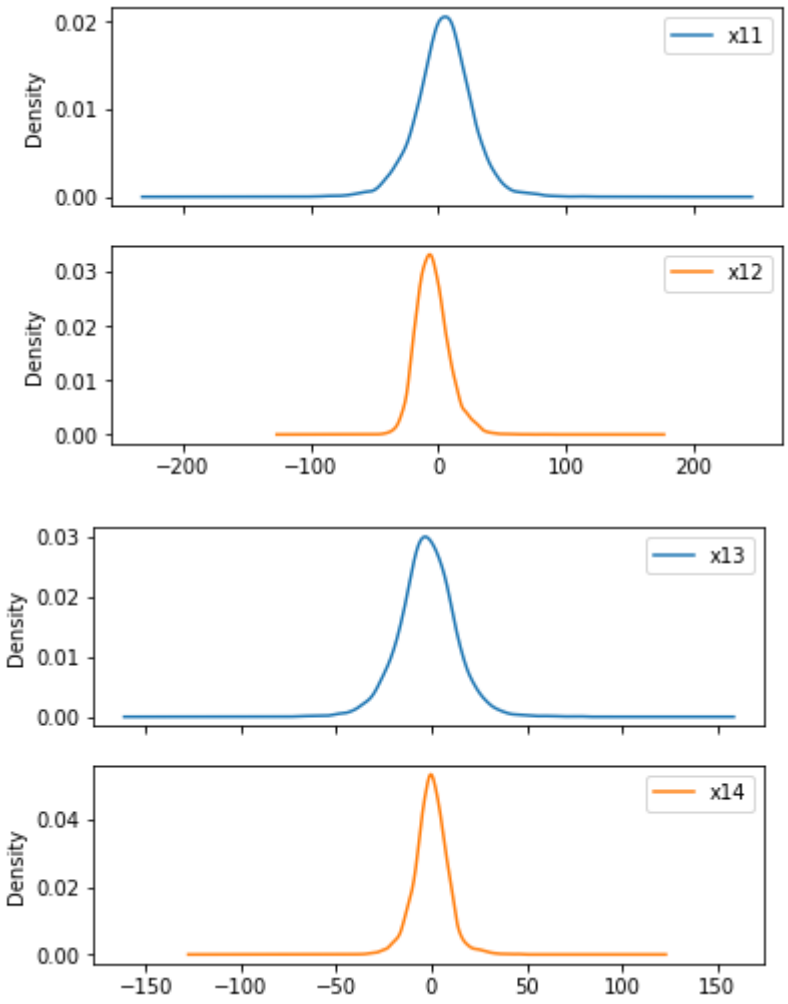
x6
Density
0.004
0.002
0.000
−1000
0
1000
2000
3000
4000

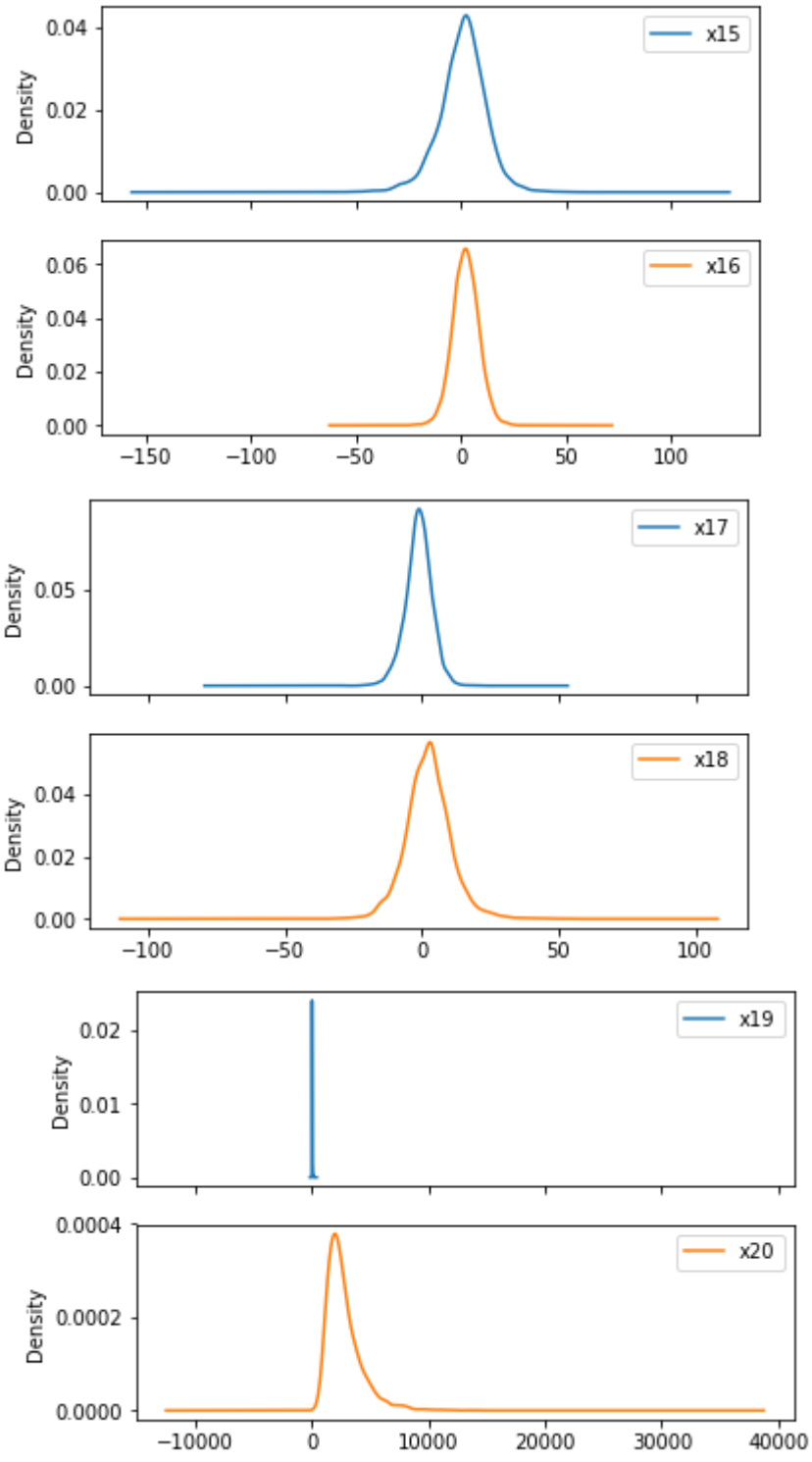

x7
Density
0.06
0.04
0.02
0.00

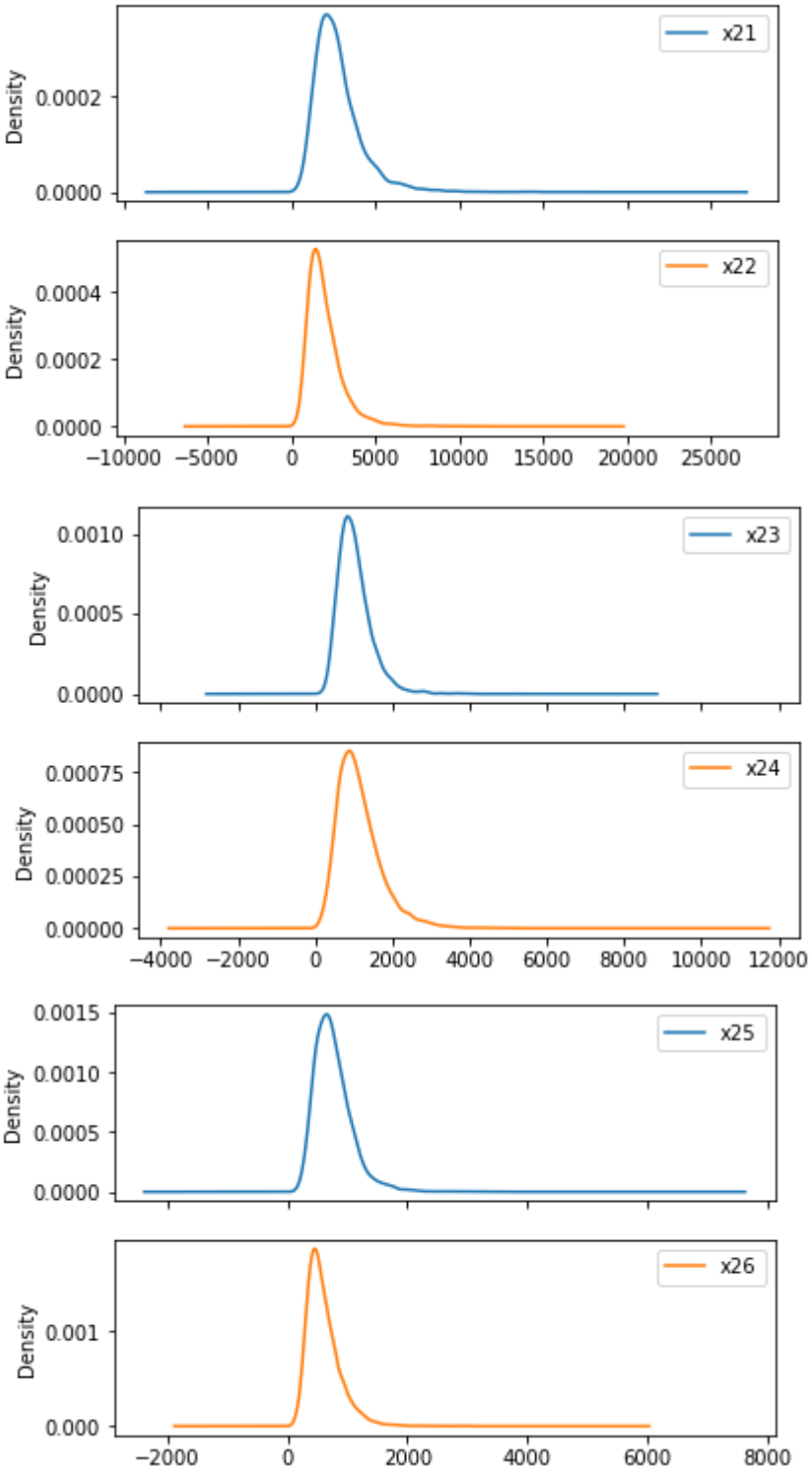

x8
Density
0.0075
0.0050
0.0025
0.0000
−600
−400
−200
0
200
400
600

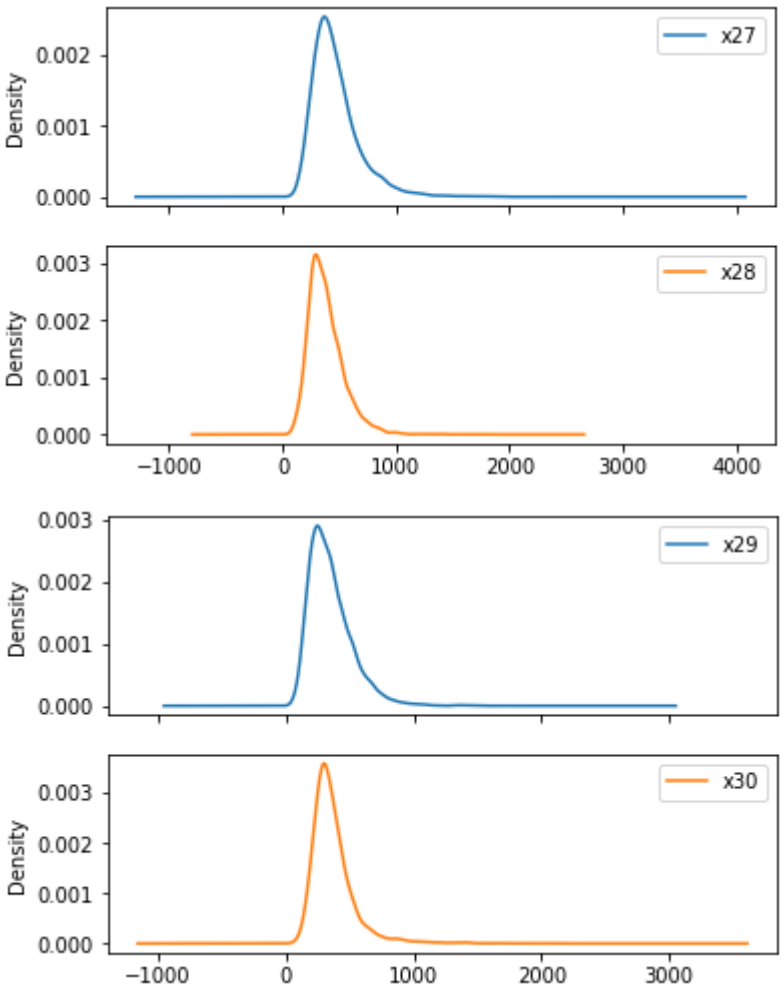

x9
Density
0.010
0.005
0.000


x10
Density
0.02
0.01
0.00
−400
−200
0
200
400


x11
Density
0.02
0.01
0.00
x12
Density
0.03
0.02
0.01
0.00
−200
−100
0
100
200
x13
Density
0.03
0.02
0.01
0.00
x14
Density
0.04
0.02
0.00
−150
−100
−50
0
50
100
150


x15
Density
0.04
0.02
0.00
x16
Density
0.06
0.04
0.02
0.00
−150
−100
−50
0
50
100
x17
Density
0.05
0.00
x18
Density
0.04
0.02
0.00
−100
−50
0
50
100
x19
Density
0.02
0.01
0.00
x20
Density
0.0004
0.0002
0.0000
−10000
0
10000
20000
30000
40000


x21
Density
0.0002
0.0000
x22
Density
0.0004
0.0002
0.0000
−10000
−5000
0
5000
10000
15000
20000
25000
x23
Density
0.0010
0.0005
0.0000
x24
Density
0.00075
0.00050
0.00025
0.00000
−4000
−2000
0
2000
4000
6000
8000
10000
12000
x25
Density
0.0015
0.0010
0.0005
0.0000
x26
Density
0.001
0.000
−2000
0
2000
4000
6000
8000


x27
Density
0.002
0.001
0.000
x28
Density
0.003
0.002
0.001
0.000
−1000
0
1000
2000
3000
4000
x29
Density
0.003
0.002
0.001
0.000
x30
Density
0.003
0.002
0.001
0.000
−1000
0
1000
2000
3000