

Trabajo de Fin de Grado

Machine Learning para el tratamiento de datos y la detección de exoplanetas mediante el método de tránsito

Resumen

Uno de los siguientes pasos en la exploración espacial es encontrar planetas más allá del sistema solar que, potencialmente, puedan albergar signos de vida extraterrestre. Estos planetas que orbitan otras estrellas son conocidos como exoplanetas. Las complejas técnicas utilizadas para su detección recaban una inmensa cantidad de datos que deben ser cuidadosamente tratados y adecuados para su posterior análisis en busca de estos mundos. Uno de estos métodos, el de tránsito, consiste en observar las estrellas en busca de disminuciones de luz provocadas por posibles exoplanetas transitando entre la estrella y el observador. Esta información queda plasmada en los datos recogidos por los telescopios, que deben ser procesados, tratados y analizados. Estas tareas se pueden llevar a cabo de forma masiva y automatizada mediante distintas técnicas de *machine learning*. En este proyecto se presentarán las principales técnicas y modelos de machine learning para el tratamiento y clasificación de datos, se emplearán algunas de ellas para adecuar conjuntos de datos de observaciones realizadas por la misión *Kepler* de la NASA y, finalmente, se construirá un modelo de predicción y se analizará su precisión a la hora de detectar exoplanetas.

```
In [82]: import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix, accuracy_score
from scipy import ndimage
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
```

Datos de entrenamiento

```
In [2]: kepler_train_data_file_path = "../TFG/datos/exoTrain.csv"
kepler_train_data = pd.read_csv(kepler_train_data_file_path)

pd.DataFrame(kepler_train_data)
```

Out[2]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.3188	FLUX.3189
0	2	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89	...	-78.07	-102
1	2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97	...	-3.28	-32
2	2	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	436.56	...	-71.69	13
3	2	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	312.42	...	5.71	-3
4	2	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-989.57	...	-594.37	-401
...
5082	1	-91.91	-92.97	-78.76	-97.33	-68.00	-68.24	-75.48	-49.25	-30.92	...	139.95	147
5083	1	989.75	891.01	908.53	851.83	755.11	615.78	595.77	458.87	492.84	...	-26.50	-4
5084	1	273.39	278.00	261.73	236.99	280.73	264.90	252.92	254.88	237.60	...	-26.82	-53
5085	1	3.82	2.09	-3.29	-2.88	1.66	-0.75	3.85	-0.03	3.28	...	10.86	-3
5086	1	323.28	306.36	293.16	287.67	249.89	218.30	188.86	178.93	118.93	...	71.19	0

5087 rows × 3198 columns

Datos de test

```
In [3]: kepler_test_data_file_path = "../TFG/datos/exoTest.csv"
kepler_test_data = pd.read_csv(kepler_test_data_file_path)

pd.DataFrame(kepler_test_data)
```

Out[3]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.3188	FLUX.3189
0	2	119.88	100.21	86.46	48.68	46.12	39.39	18.57	6.98	6.63	...	14.52	19.29
1	2	5736.59	5699.98	5717.16	5692.73	5663.83	5631.16	5626.39	5569.47	5550.44	...	-581.91	-984.09
2	2	844.48	817.49	770.07	675.01	605.52	499.45	440.77	362.95	207.27	...	17.82	-51.66
3	2	-826.00	-827.31	-846.12	-836.03	-745.50	-784.69	-791.22	-746.50	-709.53	...	122.34	93.03
4	2	-39.57	-15.88	-9.16	-6.37	-16.13	-24.05	-0.90	-45.20	-5.04	...	-37.87	-61.85
...
565	1	374.46	326.06	319.87	338.23	251.54	209.84	186.35	167.46	135.45	...	-123.55	-166.90
566	1	-0.36	4.96	6.25	4.20	8.26	-9.53	-10.10	-4.54	-11.55	...	-12.40	-5.99
567	1	-54.01	-44.13	-41.23	-42.82	-39.47	-24.88	-31.14	-24.71	-13.12	...	-0.73	-1.64
568	1	91.36	85.60	48.81	48.69	70.05	22.30	11.63	37.86	28.27	...	2.44	11.53
569	1	3071.19	2782.53	2608.69	2325.47	2089.37	1769.56	1421.09	1142.09	902.31	...	695.41	865.97

570 rows × 3198 columns

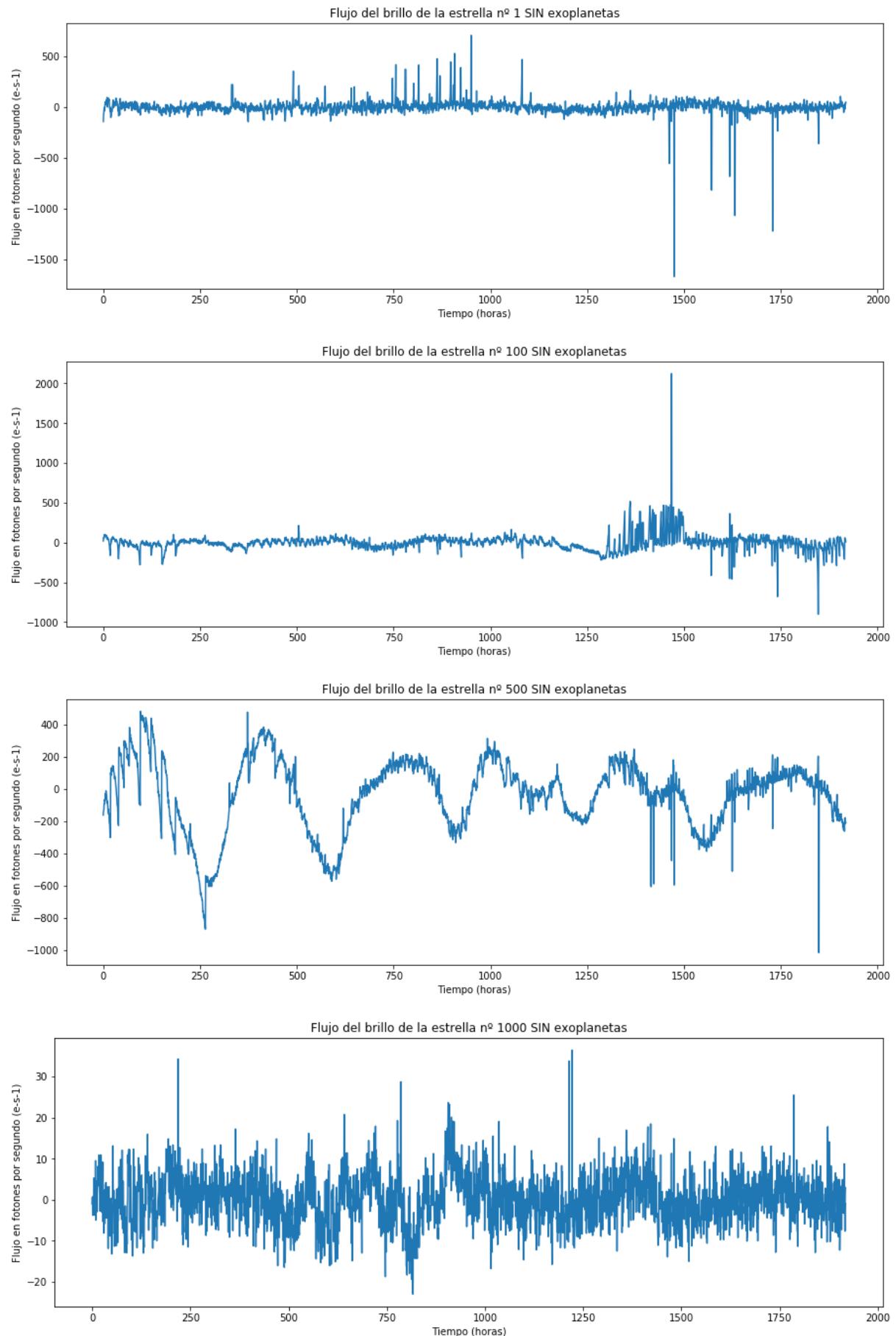


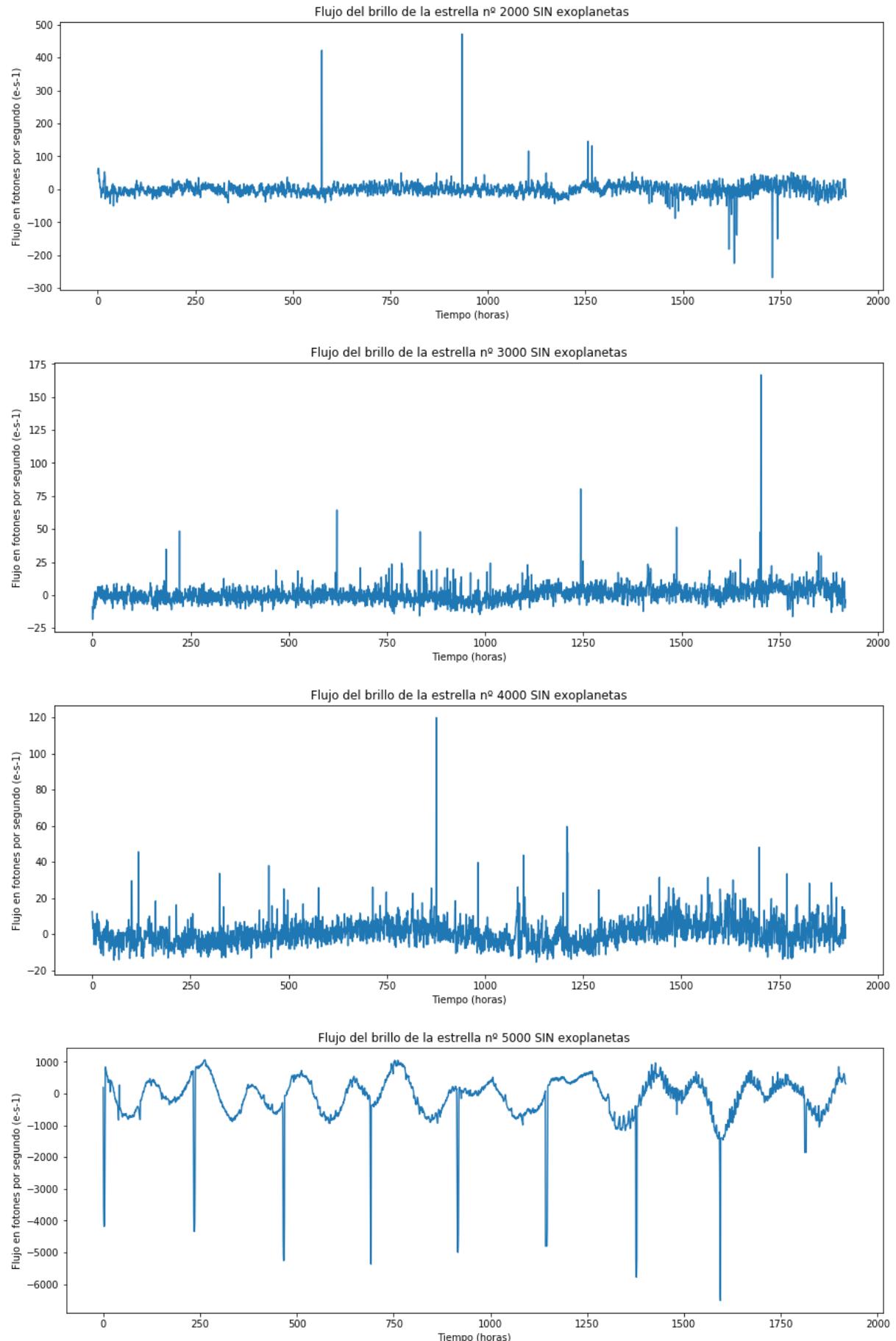
Representación Gráfica de los datos sin tratar

Diferencias entre estrellas sin exoplanetas y estrellas con exoplanetas

Estrellas sin exoplanetas ('LABEL' = 1)

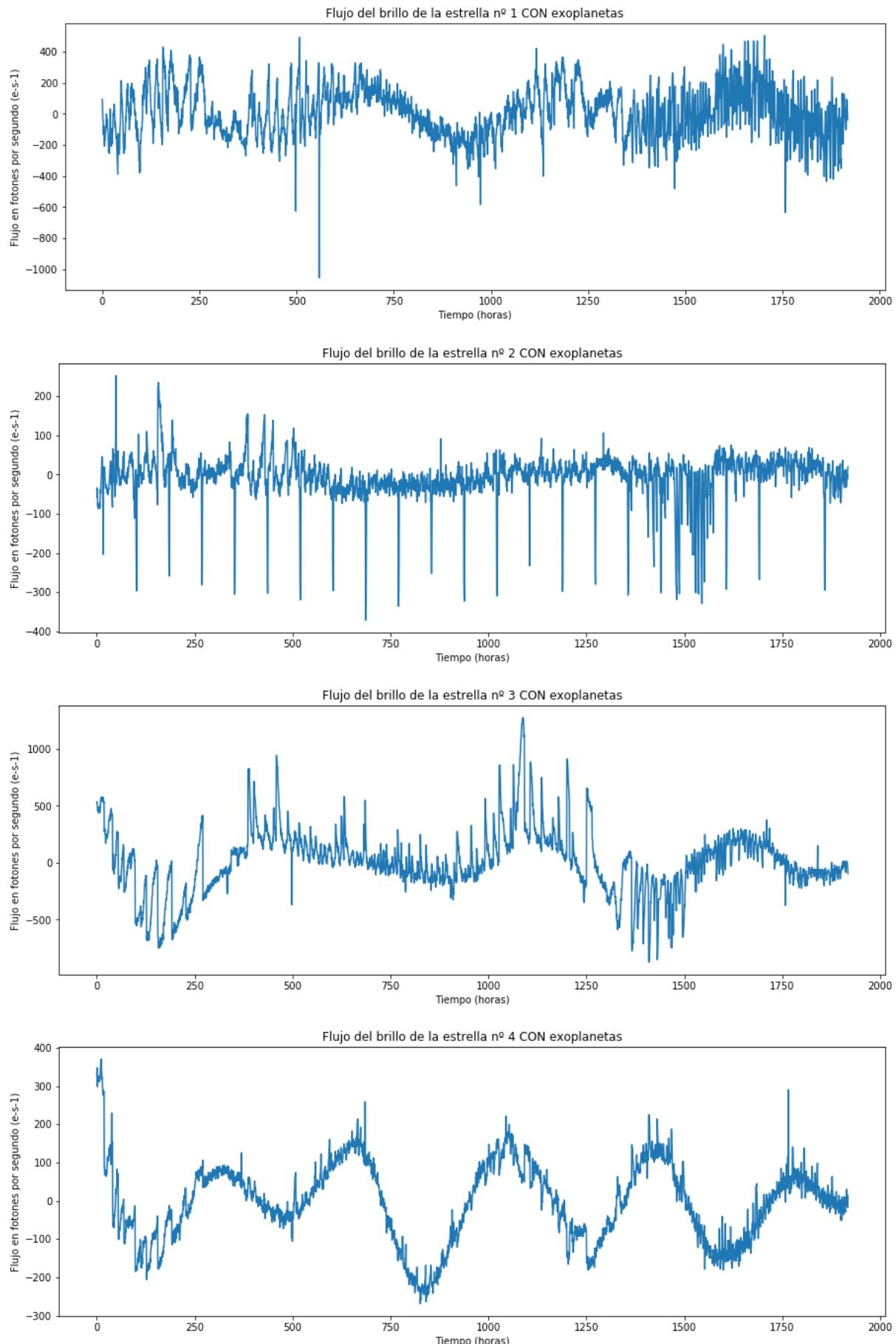
```
In [4]: for i in [0, 99, 499, 999, 1999, 2999, 3999, 4999]:  
    #Se extraen los datos de las distintas mediciones de luz, eliminando la columna "LABEL"  
    flujo = kepler_train_data[kepler_train_data.LABEL == 1].drop('LABEL', axis = 1).iloc[i,:]  
    tiempo = np.arange(len(flujo)) * (36/60) #Variable "tiempo" en horas  
    plt.figure(figsize=(15, 5)) #Tamaño del gráfico  
    plt.title('Flujo del brillo de la estrella nº {} SIN exoplanetas'.format(i+1))  
    plt.xlabel('Tiempo (horas)')  
    plt.ylabel('Flujo en fotones por segundo (e-s-1)')  
    plt.plot(tiempo, flujo)
```

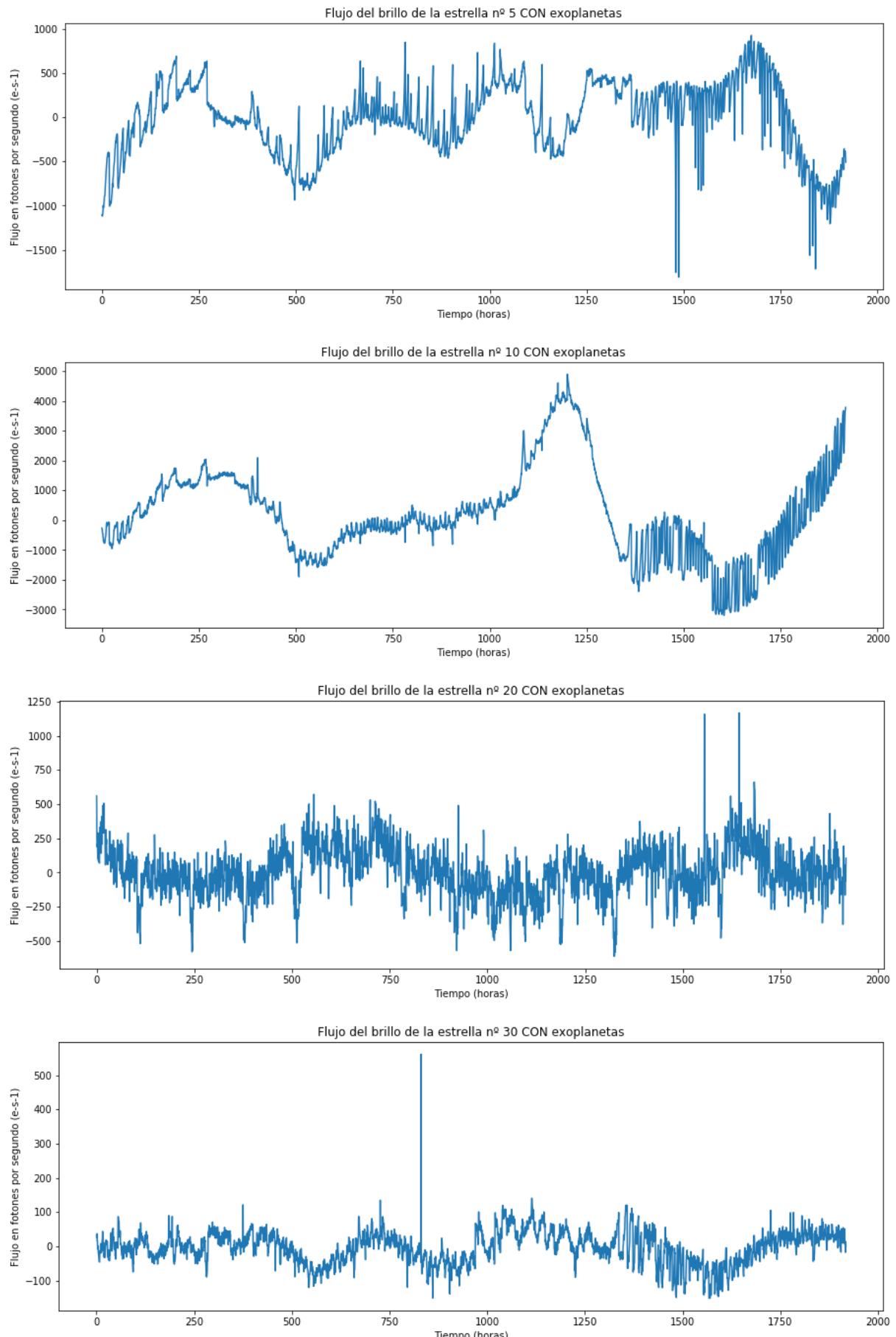




Estrellas con exoplanetas ('LABEL' = 2)

```
In [5]: for i in [0, 1, 2, 3, 4, 9, 19, 29]:
    #Se extraen los datos de las distintas mediciones de luz, eliminando la columna "LABEL"
    flujo = kepler_train_data[kepler_train_data.LABEL == 2].drop('LABEL', axis = 1).iloc[i,:]
    tiempo = np.arange(len(flujo)) * (36/60) #Variable "tiempo" en horas
    plt.figure(figsize=(15, 5)) #Tamaño del gráfico
    plt.title('Flujo del brillo de la estrella nº {} CON exoplanetas'.format(i+1))
    plt.xlabel('Tiempo (horas)')
    plt.ylabel('Flujo en fotones por segundo (e-s-1)')
    plt.plot(tiempo, flujo)
```





Tratamiento de los datos

Como se puede observar, las gráficas de los flujos de luz de las estrellas se presentan de forma muy heterogénea en cuanto a forma y magnitud de los datos, por lo que es necesario adaptarlos a un estándar para que el entrenamiento del modelo no se vea influenciado por estas diferencias.

Se va a utilizar la estrella nº 10 CON exoplanetas a modo de ejemplo del tratamiento de datos y, posteriormente, se aplicará a todo el conjunto de datos.

Abstracción del flujo

Se van a aplicar diferentes métodos para que los datos de cada estrella reflejen únicamente las variaciones de luz absolutas con el objetivo de facilitar la identificación de patrones mediante las variaciones de flujo que sí son indicativas de la presencia de exoplanetas.

Esto es necesario ya que cada estrella tiene una intensidad diferente debido a la multitud de tamaños, formas de rotación, heterogeneidad de superficies, etc. que presentan. Es por esto que cada estrella tiene una forma diferente de brillar que puede variar de múltiples formas incluso en una misma estrella. El objetivo es eliminar estas variaciones para desvincular las variaciones de flujo provocadas por posibles exoplanetas de las variaciones derivadas de las fluctuaciones de la propia estrella.

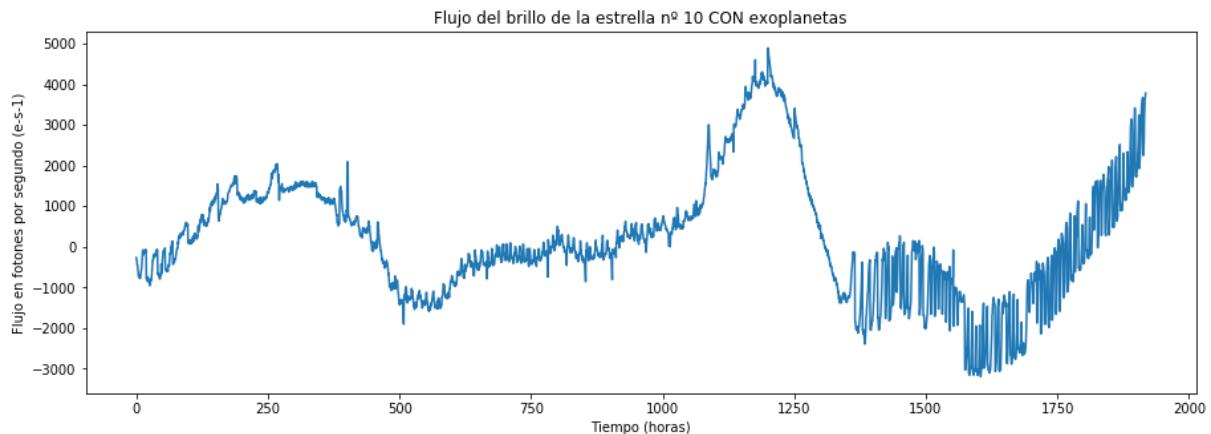
Para ello, es necesario encontrar un flujo general de los datos que se pueda sustraer del flujo original para obtener su representación plana relativa.

Primero, se va a utilizar la técnica de desenfoque gaussiano para "suavizar" los datos.

Esta es la estrella nº 10 CON exoplanetas sin ningún tipo de tratamiento.

```
In [6]: i = 9
#Se extraen los datos de las distintas mediciones de Luz, eliminando la columna "LABEL"
flujo1 = kepler_train_data[kepler_train_data.LABEL == 2].drop('LABEL', axis = 1).iloc[i,:]
tiempo = np.arange(len(flujo1)) * (36/60) #Variable "tiempo" en horas
plt.figure(figsize=(15, 5)) #Tamaño del gráfico
#Etiquetas
plt.title('Flujo del brillo de la estrella nº {} CON exoplanetas'.format(i+1))
plt.xlabel('Tiempo (horas)')
plt.ylabel('Flujo en fotones por segundo (e-s-1)')
plt.plot(tiempo, flujo1)
```

Out[6]: [`<matplotlib.lines.Line2D at 0x2cc8817cf8>`]

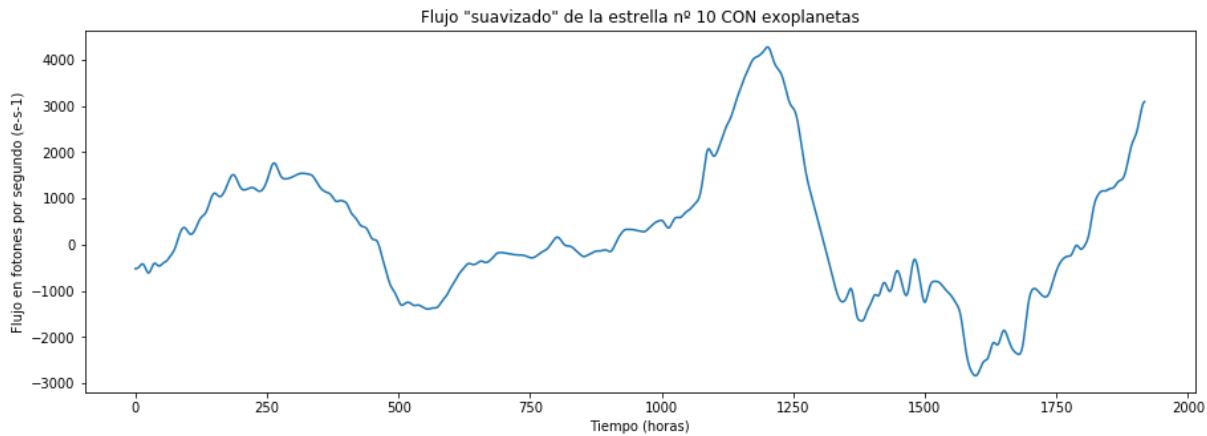


1. Desenfoque gaussiano.

Se utiliza este método para abstraer el flujo de fluctuaciones propias de la estrella

```
In [7]: flujo2 = ndimage.filters.gaussian_filter(flujo1, sigma = 10) #Se crea la variable 'flujo2' con el resultado de aplicar el desenfoque gaussiano con sigma = 10
tiempo = np.arange(len(flujo2)) * (36/60) #Variable "tiempo" en horas
plt.figure(figsize=(15, 5)) #Tamaño del gráfico
#Etiquetas
plt.title('Flujo "suavizado" de la estrella nº {} CON exoplanetas'.format(i+1))
plt.xlabel('Tiempo (horas)')
plt.ylabel('Flujo en fotones por segundo (e-s-1)')
plt.plot(tiempo, flujo2)
```

Out[7]: [<matplotlib.lines.Line2D at 0x2cc89374b48>]

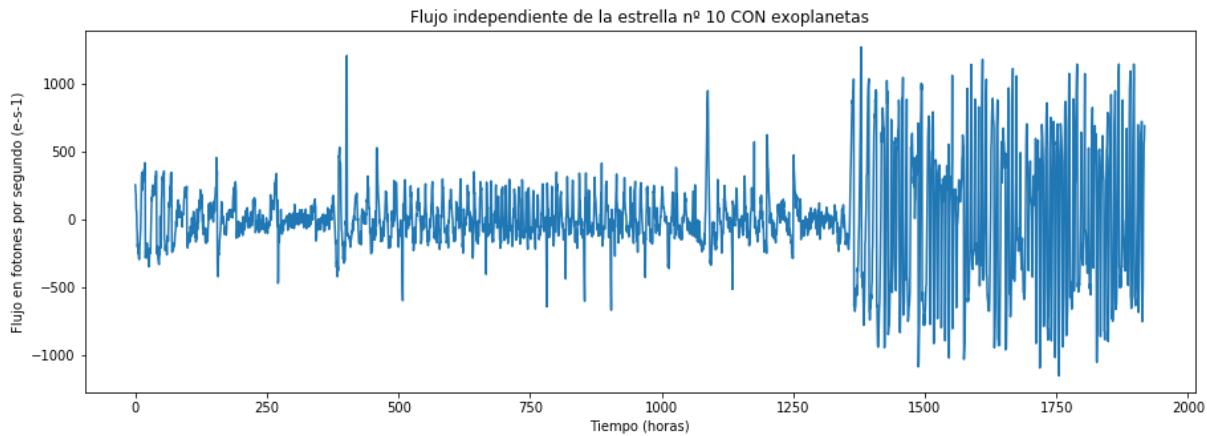


2. Desvinculación de la tendencia de fluctuaciones de la estrella.

Para abstraer esta información, se restan el flujo de datos original y el flujo de datos resultante del desenfoque gaussiano. Este nuevo flujo reflejará las variaciones de luz independientes de la tendencia de flujo de la estrella para mostrar aquellas variaciones que son relevantes para la detección de exoplanetas

```
In [8]: flujo3 = flujo1 - flujo2
tiempo = np.arange(len(flujo3)) * (36/60) #Variable "tiempo" en horas
plt.figure(figsize=(15, 5)) #Tamaño del gráfico
#Etiquetas
plt.title('Flujo independiente de la estrella nº {} CON exoplanetas'.format(i+1))
plt.xlabel('Tiempo (horas)')
plt.ylabel('Flujo en fotones por segundo (e-s-1)')
plt.plot(tiempo, flujo3)
```

Out[8]: [<matplotlib.lines.Line2D at 0x2cc894584c8>]

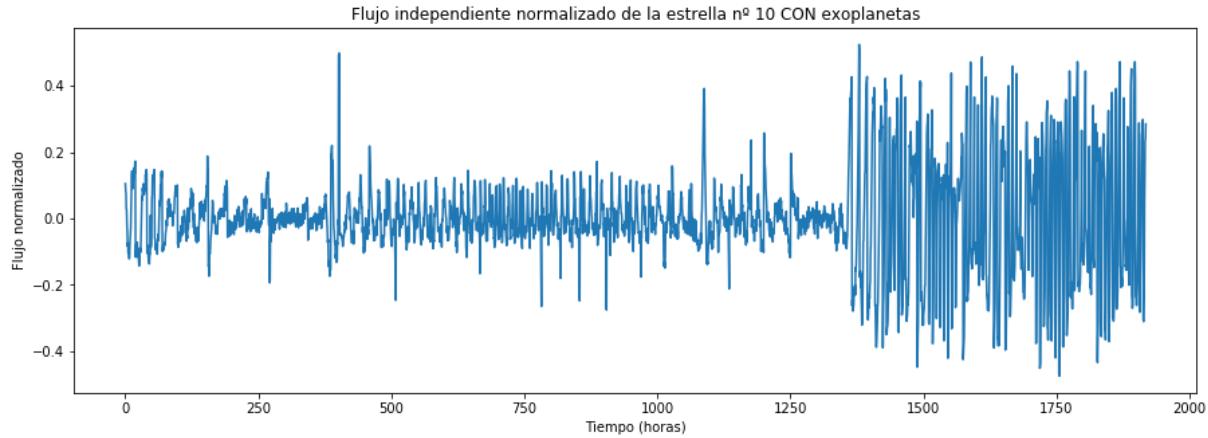


3. Normalización

El siguiente paso es normalizar el flujo de datos

```
In [9]: flujo3n = (flujo3 - np.mean(flujo3)) / (np.max(flujo3) - np.min(flujo3))
tiempo = np.arange(len(flujo3n)) * (36/60) #Variable "tiempo" en horas
plt.figure(figsize=(15, 5)) #Tamaño del gráfico
#Etiquetas
plt.title('Flujo independiente normalizado de la estrella nº {} CON exoplanetas'.format(i+1))
plt.xlabel('Tiempo (horas)')
plt.ylabel('Flujo normalizado')
plt.plot(tiempo, flujo3n)
```

Out[9]: [<matplotlib.lines.Line2D at 0x2cc894c8788>]



4. Eliminar datos atípicos superiores

Los exoplanetas provocan una disminución de la luz percibida de la estrella al transitar delante de ella, por lo que es necesario eliminar todos los datos atípicos superiores, ya que son los inferiores los que son relevantes.

```
In [10]: def eliminar_datos_atipicos_superiores(X, reducir = 0.01, amplitud=4):
    #https://www.kaggle.com/muonneutrino/exoplanet-data-visualization-and-exploration
    longitud = len(X.iloc[0,:])
    eliminar = int(longitud*reducir)
    for i in X.index.values:
        valores = X.loc[i,:]
        valores_ordenados = valores.sort_values(ascending = False)
        for j in range(eliminar):
            idx = valores_ordenados.index[j]
            #print(idx)
            nuevo_valor = 0
            c = 0
            idx_num = int(idx[5:])
            for k in range(2*amplitud+1):
                idx2 = idx_num + k - amplitud
                if idx2 < 1 or idx2 >= longitud or idx_num == idx2:
                    continue
                nuevo_valor += valores['FLUX.'+str(idx2)]
            c += 1
            nuevo_valor /= c
            if nuevo_valor < valores[idx]:
                X.at[i,idx] = nuevo_valor

    return X
```

Aplicar este proceso a todo el *dataset* de entrenamiento y test

Crear una función para aplicar el desenfoque Gaussiano, la abstracción de flujo y la normalización de los datos

```
In [11]: def abstraer_normalizar_flujo(X):
    flujo1 = X
    flujo2 = ndimage.filters.gaussian_filter(flujo1, sigma = 10) #Desenfoque Gaussiano
    flujo3 = flujo1 - flujo2 #Abstraccion de flujo
    flujo3n = (flujo3 - np.mean(flujo3)) / (np.max(flujo3) - np.min(flujo3)) #Normalizacion
    return flujo3n
```

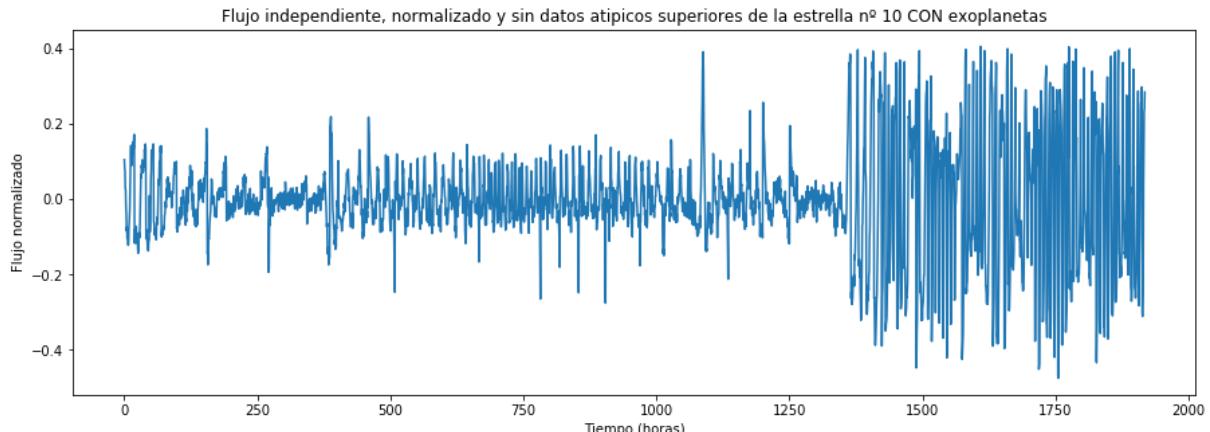
Aplicar los pasos anteriores a todos los datos tanto de entrenamiento como de test

```
In [12]: kepler_train_data.iloc[:,1:] = kepler_train_data.iloc[:,1:].apply(abstraer_normalizar_flujo, axis=1)
#Aplicar pasos 1, 2 y 3 en datos de entrenamiento
kepler_test_data.iloc[:,1:] = kepler_test_data.iloc[:,1:].apply(abstraer_normalizar_flujo, axis=1)
#Aplicar pasos 1, 2 y 3 en datos de test
kepler_train_data.iloc[:,1:] = eliminar_datos_atipicos_superiores(kepler_train_data.iloc[:,1:])#Aplicar paso 4 en datos de entrenamiento
kepler_test_data.iloc[:,1:] = eliminar_datos_atipicos_superiores(kepler_test_data.iloc[:,1:])#Aplicar paso 4 en datos de test
```

Analizar el resultado de este tratamiento con la estrella nº 10 con exoplanetas

```
In [13]: i = 9
#Se extraen los datos de las distintas mediciones de luz, eliminando la columna "LABEL"
flujo = kepler_train_data[kepler_train_data.LABEL == 2].drop('LABEL', axis = 1).iloc[i,:]
tiempo = np.arange(len(flujo)) * (36/60) #Variable "tiempo" en horas
plt.figure(figsize=(15, 5)) #Tamaño del gráfico
#Etiquetas
plt.title('Flujo independiente, normalizado y sin datos atípicos superiores de la estrella nº {} CON exoplanetas'.format(i+1))
plt.xlabel('Tiempo (horas)')
plt.ylabel('Flujo normalizado')
plt.plot(tiempo, flujo)
```

Out[13]: [<matplotlib.lines.Line2D at 0x2cc89471c48>]



Visualización de los datos

Así quedan los datos después del primer tratamiento

Datos de entrenamiento

In [14]: `pd.DataFrame(kepler_train_data)`

Out[14]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.31
0	2	0.106403	0.100007	0.058134	0.027612	0.020326	-0.035103	-0.040581	-0.013007	-0.000332	...	-0.046101
1	2	0.041485	0.050703	0.003597	0.037446	-0.037595	-0.026886	-0.054623	-0.054545	-0.054560	...	0.007100
2	2	0.036390	0.039271	0.018812	0.003153	-0.034190	-0.026011	-0.027999	-0.008730	-0.054926	...	-0.036100
3	2	0.019349	0.081422	-0.052651	-0.065242	-0.006866	-0.021728	0.007249	-0.024948	-0.020675	...	0.038100
4	2	-0.058758	-0.061891	-0.066326	-0.059138	-0.047121	-0.041884	-0.031973	-0.047753	-0.040327	...	-0.038100
...
5082	1	-0.117483	-0.121318	-0.095393	-0.138459	-0.084585	-0.092969	-0.117113	-0.073702	-0.047662	...	0.173100
5083	1	0.234930	0.184672	0.197383	0.172634	0.128328	0.062953	0.061642	-0.000242	0.028688	...	-0.020100
5084	1	0.018139	0.020752	0.012979	0.001144	0.024486	0.017697	0.013113	0.015868	0.009033	...	-0.025100
5085	1	0.007295	0.004060	-0.006046	-0.005157	0.003582	-0.000798	0.008118	0.001014	0.007540	...	-0.002100
5086	1	0.088057	0.075981	0.067276	0.064914	0.038915	0.018064	-0.000705	-0.004434	-0.045276	...	0.048100

5087 rows × 3198 columns



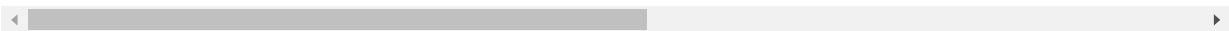
Datos de test

In [15]: `pd.DataFrame(kepler_test_data)`

Out[15]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.31
0	2	0.112855	0.193746	0.156624	0.053047	0.048003	0.031599	-0.023958	-0.053856	-0.052662	...	-0.067100
1	2	0.040047	0.038188	0.041250	0.042059	0.043381	0.045242	0.049985	0.051557	0.056664	...	0.019100
2	2	0.411868	0.346021	0.291078	0.248623	0.296998	0.187838	0.138318	0.068238	-0.093263	...	0.052100
3	2	-0.024040	-0.024718	-0.030960	-0.028750	-0.002513	-0.015710	-0.019292	-0.007635	0.001454	...	0.067100
4	2	-0.012001	-0.001298	0.001697	0.002894	-0.001611	-0.005295	0.005089	-0.015073	0.003032	...	0.012100
...
565	1	0.076486	0.055972	0.054183	0.063447	0.027594	0.011578	0.003751	-0.001835	-0.012910	...	0.017100
566	1	0.021322	0.032763	0.036279	0.033286	0.043332	0.008502	0.009654	0.023791	0.012153	...	-0.013100
567	1	-0.108557	-0.070560	-0.060486	-0.068568	-0.057884	-0.003740	-0.031567	-0.010125	0.031219	...	-0.004100
568	1	0.012061	0.010948	0.003639	0.003743	0.008203	-0.001205	-0.003133	0.002375	0.000689	...	0.000100
569	1	0.043405	0.036265	0.049002	0.038708	0.030668	0.019543	0.007517	-0.001485	-0.008744	...	0.015100

570 rows × 3198 columns



PCA

Con los datos ya tratados, el siguiente paso es aplicar PCA para la reducción de dimensiones.

Instanciar y aplicar PCA en el dataset de entrenamiento

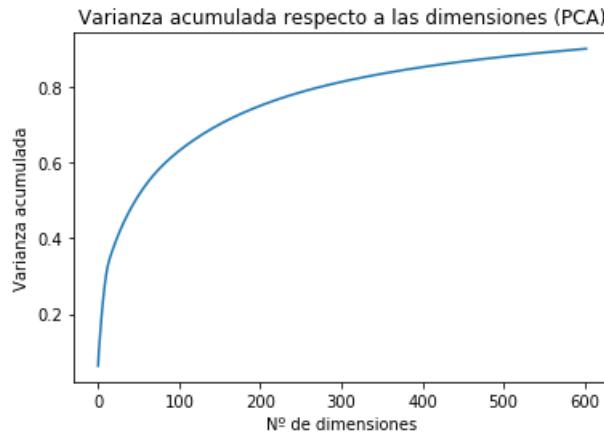
```
In [16]: flujos = kepler_train_data.drop(['LABEL'], axis = 1) #Eliminar la columna etiqueta
pca = PCA(0.9) #Crear una instancia de PCA con las dimensiones necesarias para obtener 0.9 de varianza
pca.fit(flujos) #Ajustar el modelo con los datos (generar matriz de covarianza, autovectores, autovalores, etc.)
flujos_pca = pca.transform(flujos) #Aplicar la reducción de dimensiones

print("Filas y dimensiones finales:", flujos_pca.shape)
expl = pca.explained_variance_ratio_
print('Suma de la varianza: ', sum(expl))

#Visualizar el grafico de la varianza acumulada
plt.title('Varianza acumulada respecto a las dimensiones (PCA)')
plt.xlabel('Nº de dimensiones')
plt.ylabel('Varianza acumulada')
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```

Filas y dimensiones finales: (5087, 603)
Suma de la varianza: 0.9001430541752459

Out[16]: [`<matplotlib.lines.Line2D at 0x2cc894c8648>`]



El resultado de *PCA* muestra que, para obtener una varianza acumulada de 0,9, son necesarias 603 dimensiones. El valor de estas dimensiones para cada estrella se cargan en "*flujos_pca*" ordenadas de mayor a menos varianza -es decir, de mayor a menor relevancia para el modelo-.

Crear el nuevo dataset de entrenamiento

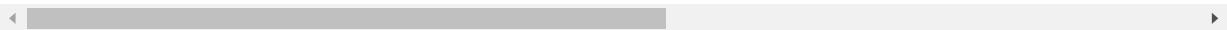
Nuevo *dataset* que contiene únicamente las 603 dimensiones resultantes de aplicar *PCA*.

```
In [17]: kepler_train_data_pca = pd.DataFrame(data = flujos_pca) #Crear dataset con los datos de 'flujo_pc a' con los valores de las 603 dimensiones
kepler_train_data_pca
```

Out[17]:

	0	1	2	3	4	5	6	7	8	9	...
0	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	-0.350702	...
1	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	-0.067823	...
2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	-0.959701	...
3	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	-0.506663	...
4	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	-0.478106	...
...
5082	-3.936119	0.222828	-0.168616	-0.652407	-0.568182	2.094444	-0.296745	-1.226294	0.457405	0.347252	...
5083	-0.091921	1.186475	1.311370	0.323385	-0.594974	-0.652316	0.471931	1.509093	1.447173	0.929856	...
5084	0.114565	0.210029	0.394792	0.090145	0.045819	-0.217128	-0.025548	0.043999	-0.194226	-0.088254	...
5085	-0.004102	-0.161846	0.041366	0.126275	-0.002037	-0.070416	0.018862	0.026613	0.005338	0.003834	...
5086	0.320166	-0.382206	-0.145770	-0.418198	0.047609	-0.114099	0.096102	-0.066002	0.185191	-0.290908	...

5087 rows × 603 columns



Adaptar el nuevo dataset de entrenamiento

Dar el mismo formato a este nuevo *dataset* para que se asemeje al original. Esto consiste en renombrar las columnas a '*FLUX.n*' y añadir la columna '*LABEL*' con los mismos valores del *dataset* original.

```
In [18]: label = [] #Crear la Lista en la que se cargaran los valores de la columna 'LABEL'
for i in range(len(kepler_train_data_pca.columns)):
    kepler_train_data_pca=kepler_train_data_pca.rename({i: 'FLUX.{}'.format(i+1)}, axis = 1) #Renombrar la i-esima columna

for i in range(len(kepler_train_data)):
    label.append(int(kepler_train_data.iloc[i]['LABEL'])) #Añadir el valor i-esimo de la columna 'LABEL' a la lista

kepler_train_data_pca.insert(0, 'LABEL', label, True) #Insertar en el nuevo dataset de entrenamiento la columna 'LABEL' con sus valores correspondientes
kepler_train_data_pca
```

Out[18]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.603
0	2	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	...	0.1564
1	2	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	...	-0.0492
2	2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	...	-0.0359
3	2	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	...	0.0278
4	2	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	...	-0.0850
...
5082	1	-3.936119	0.222828	-0.168616	-0.652407	-0.568182	2.094444	-0.296745	-1.226294	0.457405	...	-0.0183
5083	1	-0.091921	1.186475	1.311370	0.323385	-0.594974	-0.652316	0.471931	1.509093	1.447173	...	-0.0575
5084	1	0.114565	0.210029	0.394792	0.090145	0.045819	-0.217128	-0.025548	0.043999	-0.194226	...	-0.0175
5085	1	-0.004102	-0.161846	0.041366	0.126275	-0.002037	-0.070416	0.018862	0.026613	0.005338	...	0.0002
5086	1	0.320166	-0.382206	-0.145770	-0.418198	0.047609	-0.114099	0.096102	-0.066002	0.185191	...	-0.0393

5087 rows × 604 columns



Arriba se muestra el nuevo *dataset* de entrenamiento con las 603 dimensiones.

Aplicar al dataset de test

Se aplica la misma función de PCA creada con los datos de entrenamiento para seleccionar las mismas dimensiones de los datos de test. Es importante seleccionar las mismas para que el modelo final evalúe los datos de test con las mismas dimensiones que tienen los datos con los que ha sido entrenado.

```
In [19]: flujos_test = kepler_test_data.drop(['LABEL'], axis = 1) #Eliminar la columna etiqueta
flujos_test_pca = pca.transform(flujos_test) #Aplicar la reducción de dimensiones con la misma instancia de PCA creada con los datos de entrenamiento

kepler_test_data_pca = pd.DataFrame(data = flujos_test_pca) #Crear dataset con los datos de 'flujos_test_pca' con los valores de las 603 dimensiones

label = [] #Crear la lista en la que se cargaran los valores de la columna 'LABEL'
for i in range(len(kepler_test_data_pca.columns)):
    kepler_test_data_pca=kepler_test_data_pca.rename({i: 'FLUX.{}'.format(i+1)}, axis = 1) #Renombrar la i-esima columna

for i in range(len(kepler_test_data)):
    label.append(int(kepler_test_data.iloc[i]['LABEL'])) #Añadir el valor i-esimo de la columna 'LABEL' a la lista

kepler_test_data_pca.insert(0, 'LABEL', label) #Insertar en el nuevo dataset de test la columna 'LABEL' con sus valores correspondientes
kepler_test_data_pca
```

Out[19]:

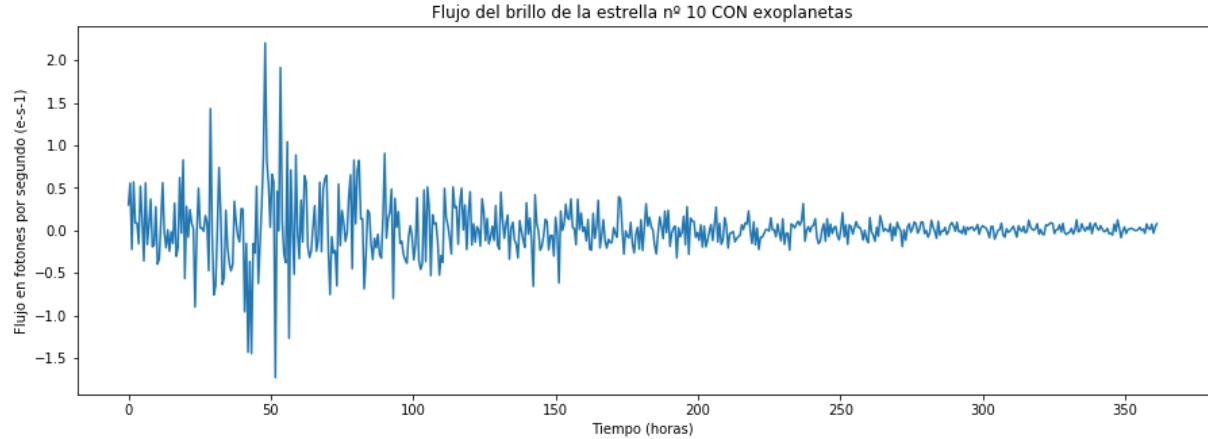
	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.59
0	2	0.065266	0.076898	0.022723	-0.027709	-0.038500	0.102257	-0.000655	0.280757	0.100139	...	-0.01364
1	2	0.155799	-0.156272	0.102767	-0.051380	0.070768	-0.279328	-0.041953	-0.151652	0.088318	...	-0.02139
2	2	0.193657	-0.242835	0.305769	-0.117894	0.017249	0.184005	0.201150	0.078778	0.140655	...	0.16050
3	2	-0.084775	-0.030296	0.263216	-0.128419	0.239384	-0.133651	-0.265096	-0.098542	0.307245	...	-0.21118
4	2	0.065127	0.003941	0.103341	0.036462	-0.062291	-0.056169	-0.046053	0.203867	0.065525	...	0.03725
...
565	1	0.008058	0.151563	-0.142312	-0.137301	0.111794	-0.087730	0.544112	0.397374	-0.497785	...	-0.05994
566	1	0.210141	-0.068512	0.230793	0.139451	0.213961	0.026154	0.147719	-0.190076	-0.383555	...	-0.00880
567	1	-0.677434	0.127153	0.223530	0.122579	0.033687	0.344718	-0.636733	-0.193548	-0.258463	...	-0.01917
568	1	0.252398	0.089392	-0.025415	-0.292439	-0.096788	0.326310	0.065421	0.158569	-0.276049	...	0.00096
569	1	0.334009	-0.288403	-0.388990	-0.122355	-0.065389	0.122939	0.088800	-0.144729	0.072246	...	-0.02021

570 rows × 604 columns

Así queda el flujo de la estrella nº 10 CON exoplanetas -del dataset de entrenamiento- con las 603 dimensiones seleccionadas por PCA.

```
In [20]: i = 9
#Se extraen los datos de las distintas mediciones de luz, eliminando la columna "LABEL"
flujo = kepler_train_data_pca[kepler_train_data_pca.LABEL == 2].drop('LABEL', axis = 1).iloc[i,:]
tiempo = np.arange(len(flujo)) * (36/60) #Variable "tiempo" en horas
plt.figure(figsize=(15, 5)) #Tamaño del gráfico
plt.title('Flujo del brillo de la estrella nº {} CON exoplanetas'.format(i+1))
plt.xlabel('Tiempo (horas)')
plt.ylabel('Flujo en fotones por segundo (e-s-1)')
plt.plot(tiempo, flujo)
```

Out[20]: [<matplotlib.lines.Line2D at 0x2cc89c51888>]



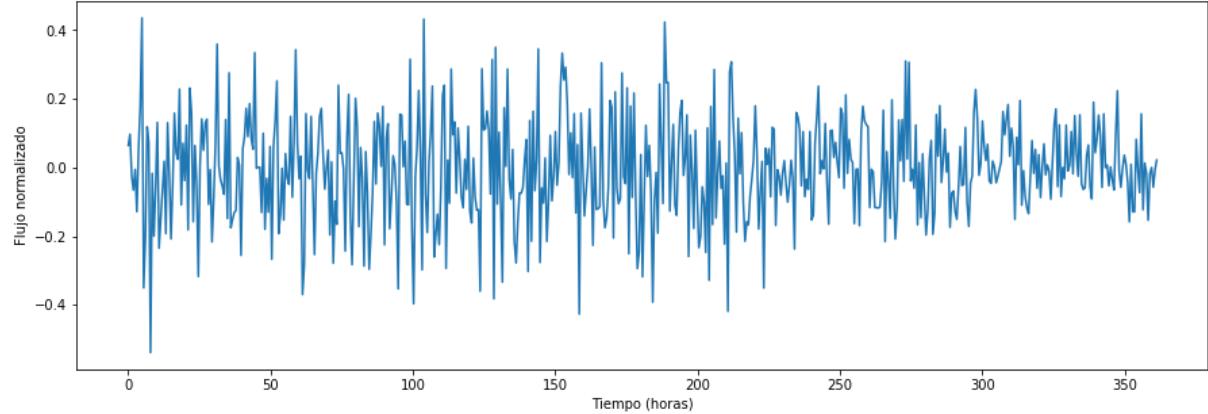
Data augmentation

Para balancear el *dataset* de entrenamiento mediante este método, es necesario conocer las características de los datos que van a ser replicados y alterados. Estos datos son los correspondientes a las estrellas con exoplanetas confirmados -*LABEL* = 2-, que son 37 de las 5.087 totales -alrededor del 0,7%-.. Por ello, es necesario obtener unos 15 nuevos registros por cada uno de los 37 existentes para lograr un mayor balanceamiento, cercano al 10%.

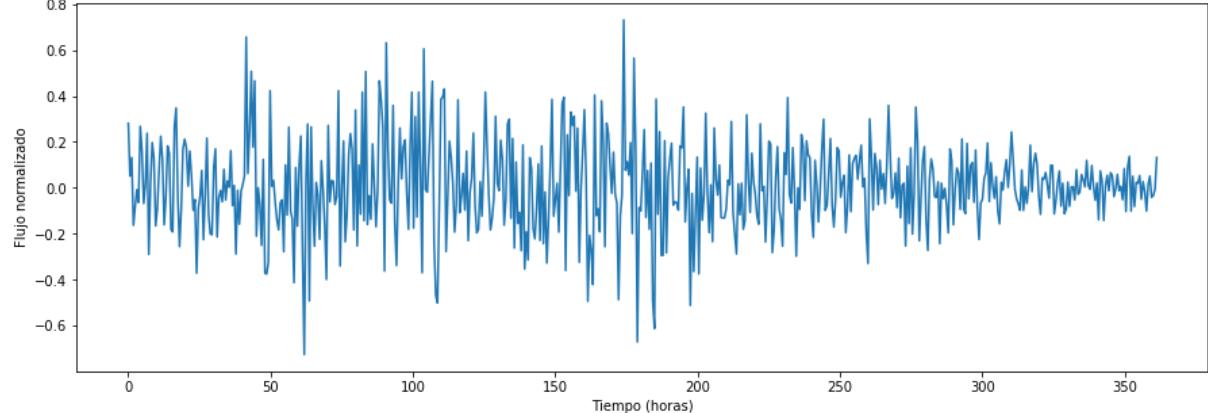
Estos son los flujos de las 37 estrellas con exoplanetas que deben ser replicados:

```
In [21]: for i in range(len(kepler_train_data_pca[kepler_train_data_pca.LABEL == 2])):
    #Se extraen los datos de las distintas mediciones de luz, eliminando la columna "LABEL"
    flujo = kepler_train_data_pca[kepler_train_data_pca.LABEL == 2].drop('LABEL', axis = 1).iloc[i,:]
    tiempo = np.arange(len(flujo)) * (36/60) #Variable "tiempo" en horas
    plt.figure(figsize=(15, 5)) #Tamaño del gráfico
    plt.title('Flujo tras aplicar PCA de la estrella nº {} CON exoplanetas'.format(i+1))
    plt.xlabel('Tiempo (horas)')
    plt.ylabel('Flujo normalizado')
    plt.plot(tiempo, flujo)
    plt.rcParams.update({'figure.max_open_warning': 0})
```

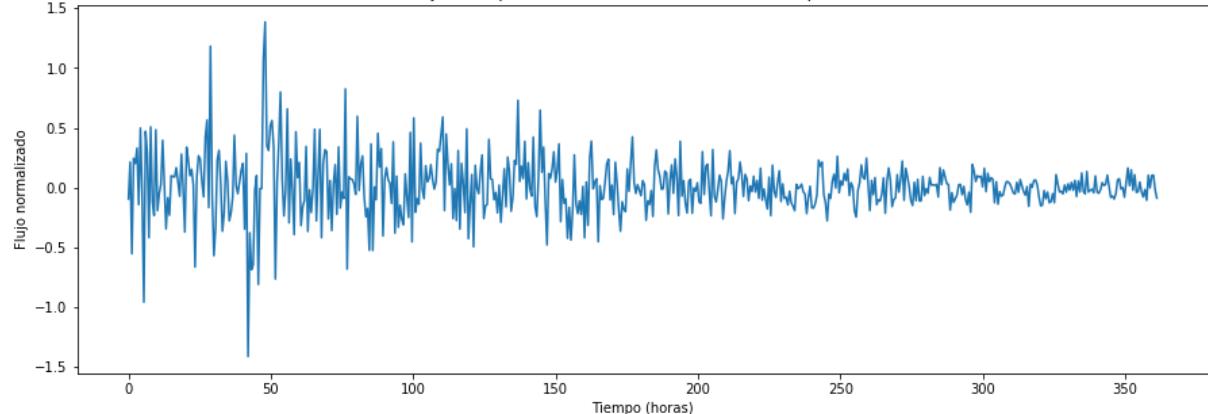
Flujo tras aplicar PCA de la estrella nº 1 CON exoplanetas



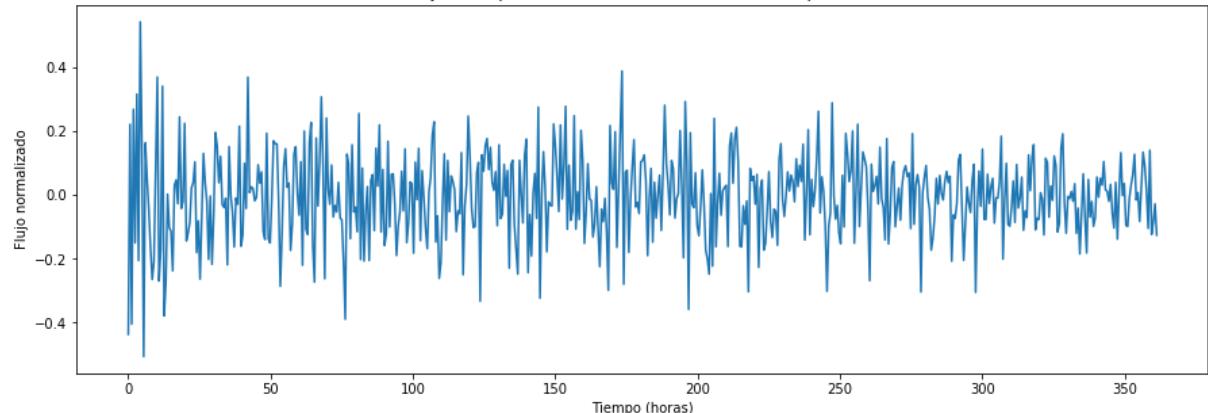
Flujo tras aplicar PCA de la estrella nº 2 CON exoplanetas



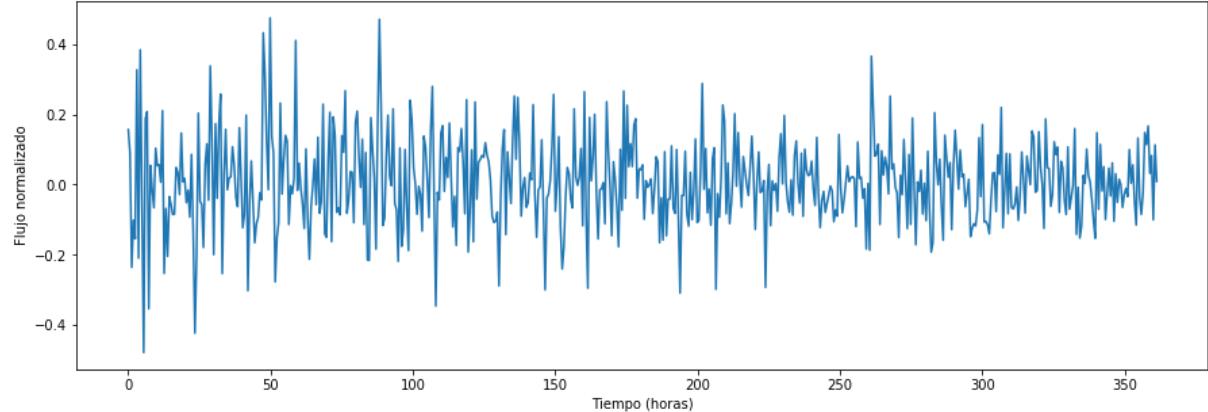
Flujo tras aplicar PCA de la estrella nº 3 CON exoplanetas



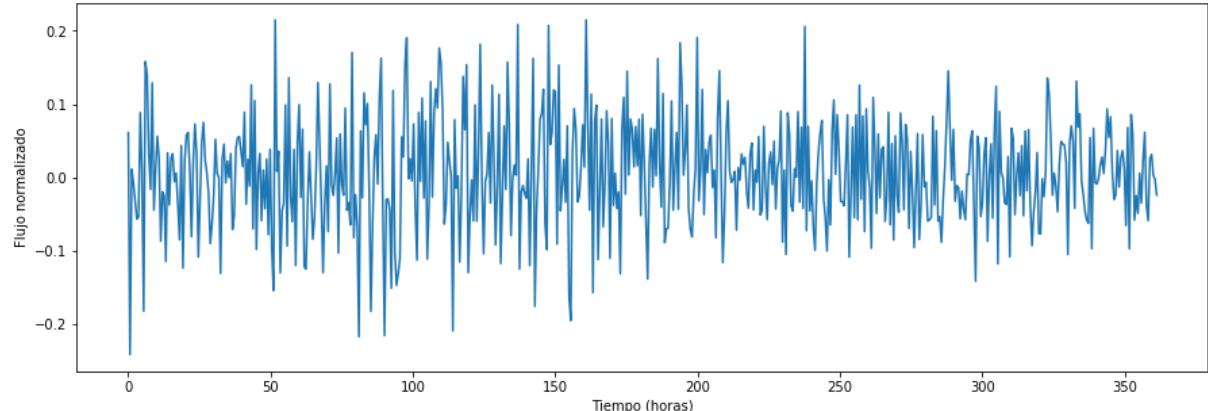
Flujo tras aplicar PCA de la estrella nº 4 CON exoplanetas



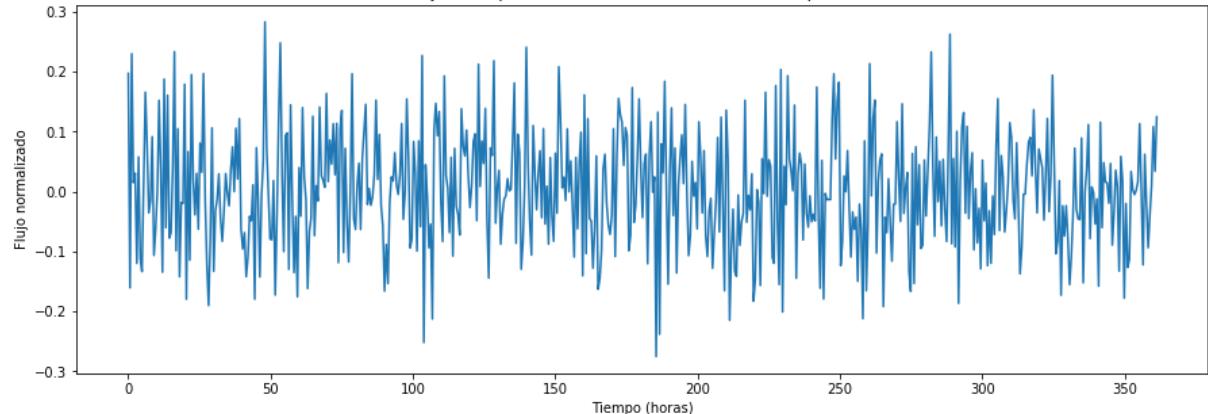
Flujo tras aplicar PCA de la estrella nº 5 CON exoplanetas



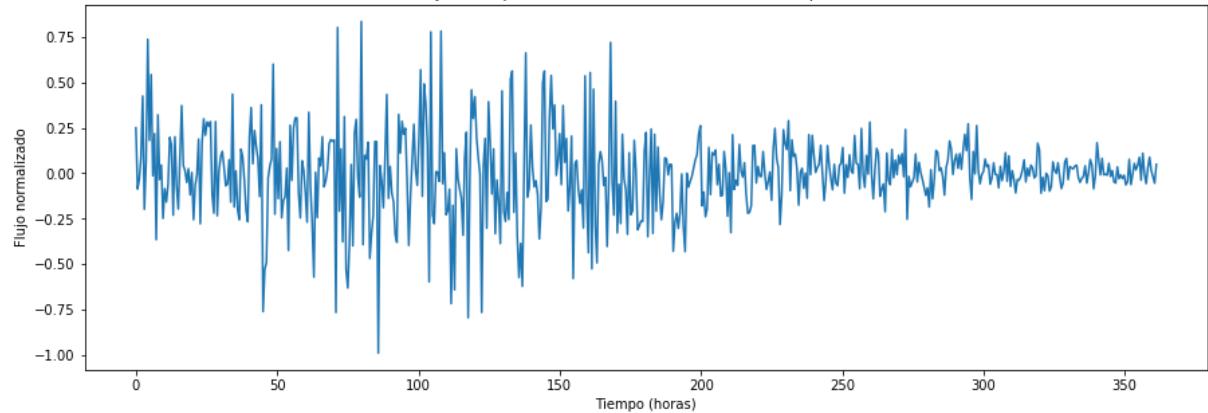
Flujo tras aplicar PCA de la estrella nº 6 CON exoplanetas



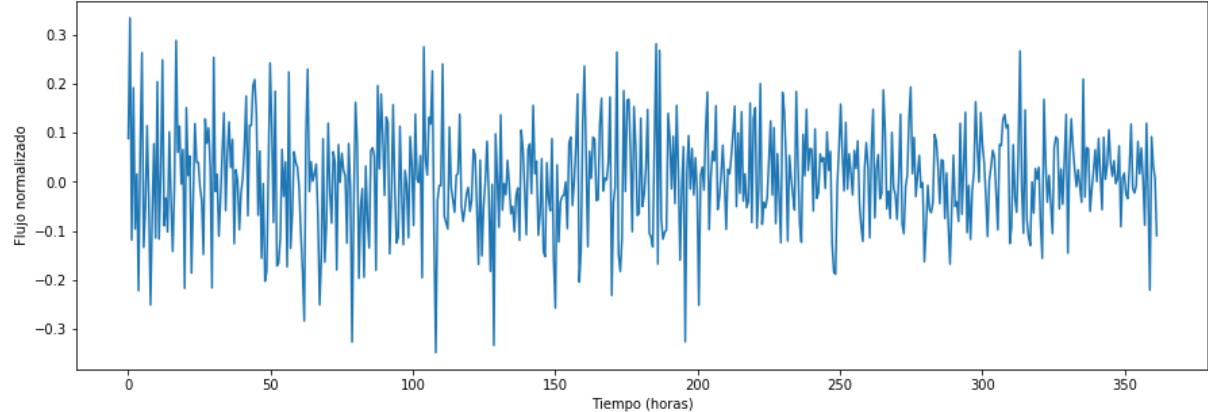
Flujo tras aplicar PCA de la estrella nº 7 CON exoplanetas



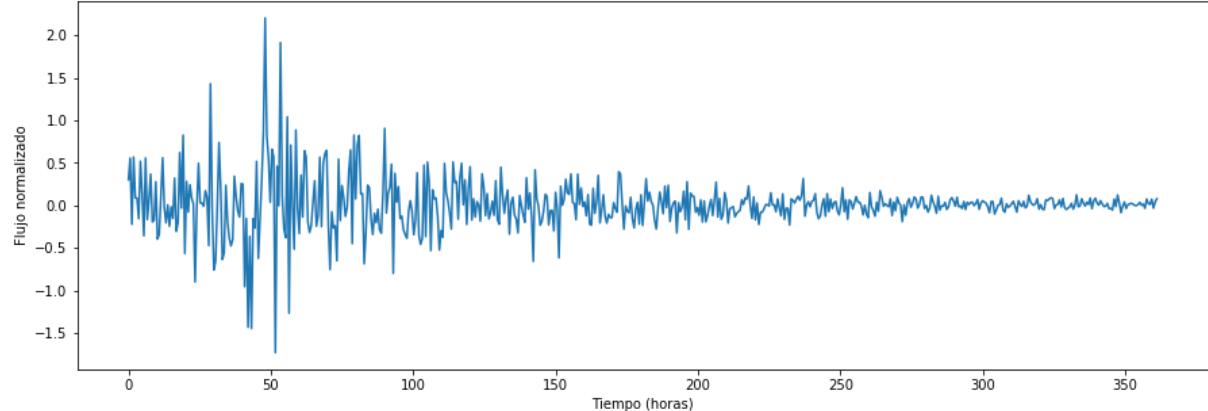
Flujo tras aplicar PCA de la estrella nº 8 CON exoplanetas



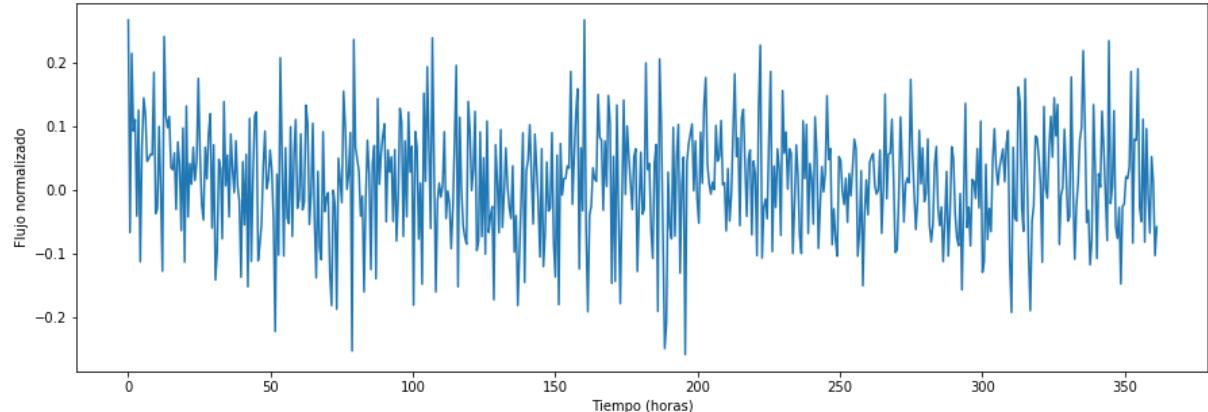
Flujo tras aplicar PCA de la estrella nº 9 CON exoplanetas



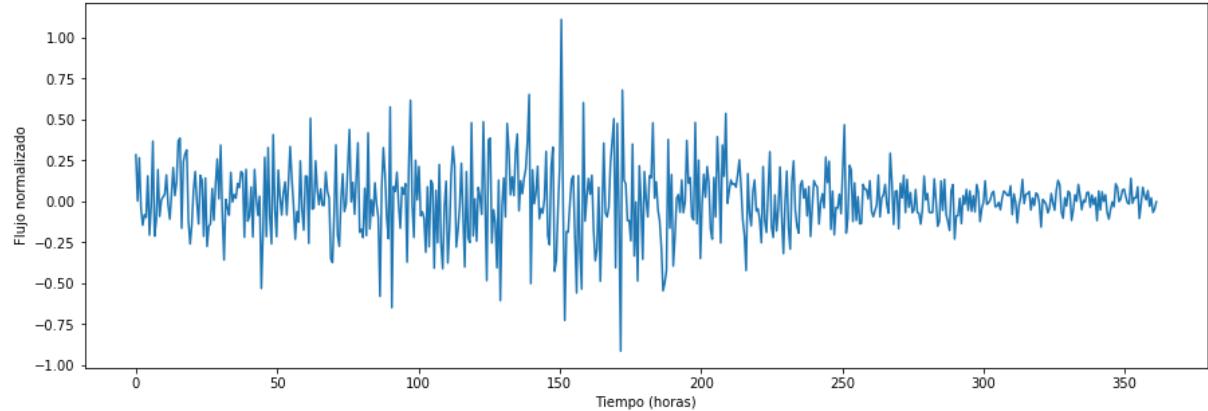
Flujo tras aplicar PCA de la estrella nº 10 CON exoplanetas



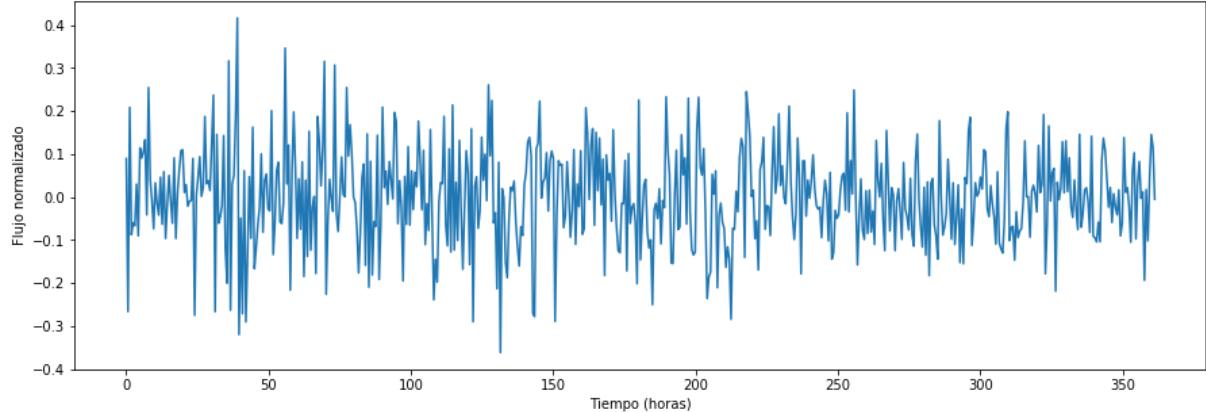
Flujo tras aplicar PCA de la estrella nº 11 CON exoplanetas



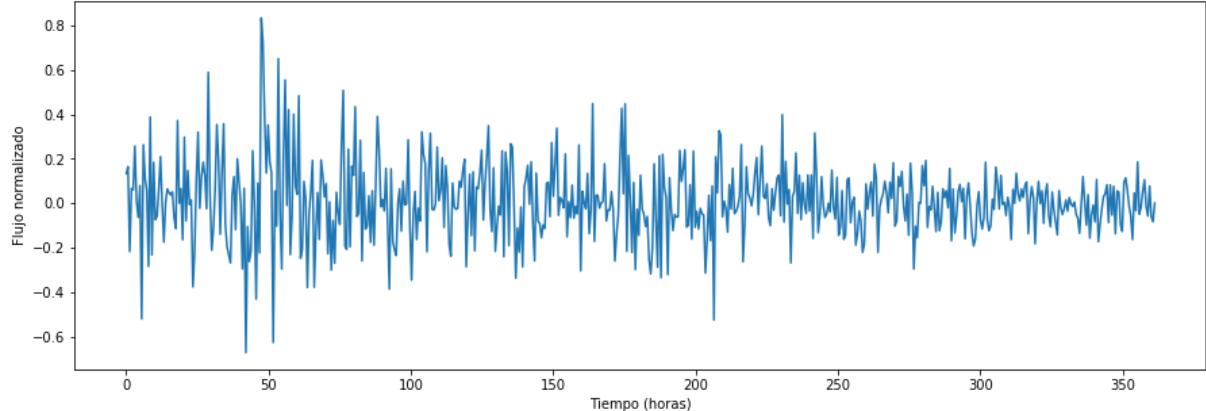
Flujo tras aplicar PCA de la estrella nº 12 CON exoplanetas



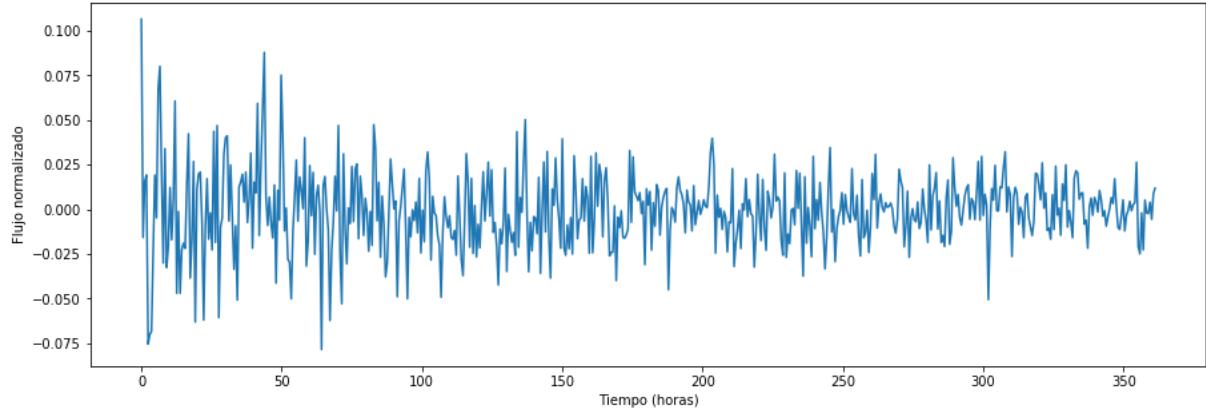
Flujo tras aplicar PCA de la estrella nº 13 CON exoplanetas



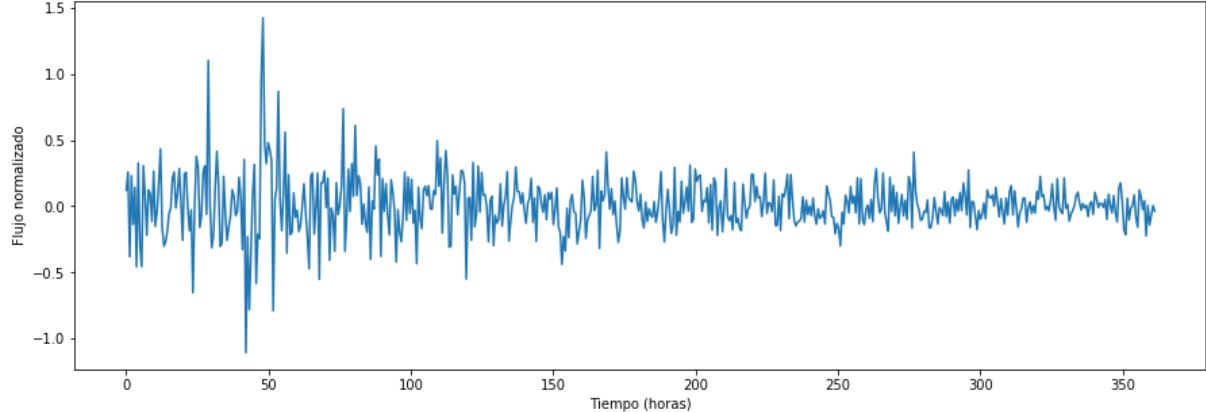
Flujo tras aplicar PCA de la estrella nº 14 CON exoplanetas



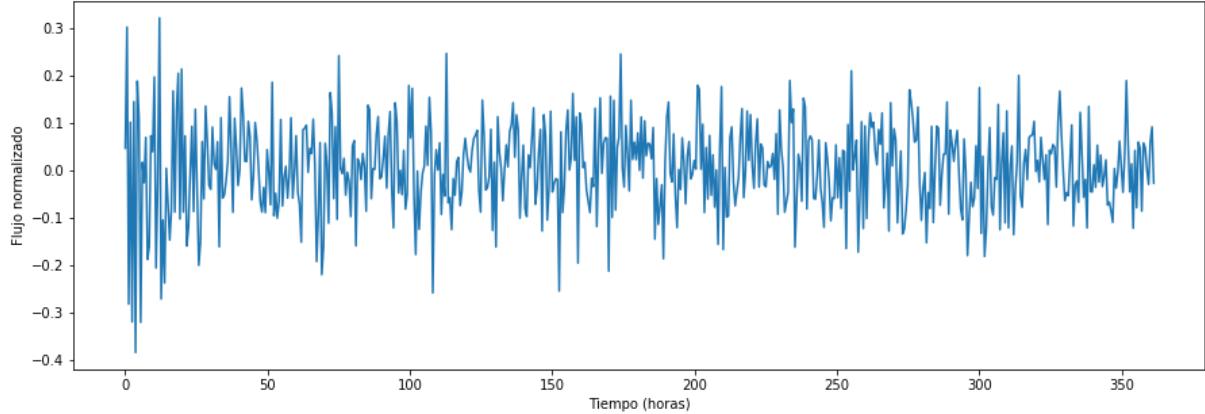
Flujo tras aplicar PCA de la estrella nº 15 CON exoplanetas



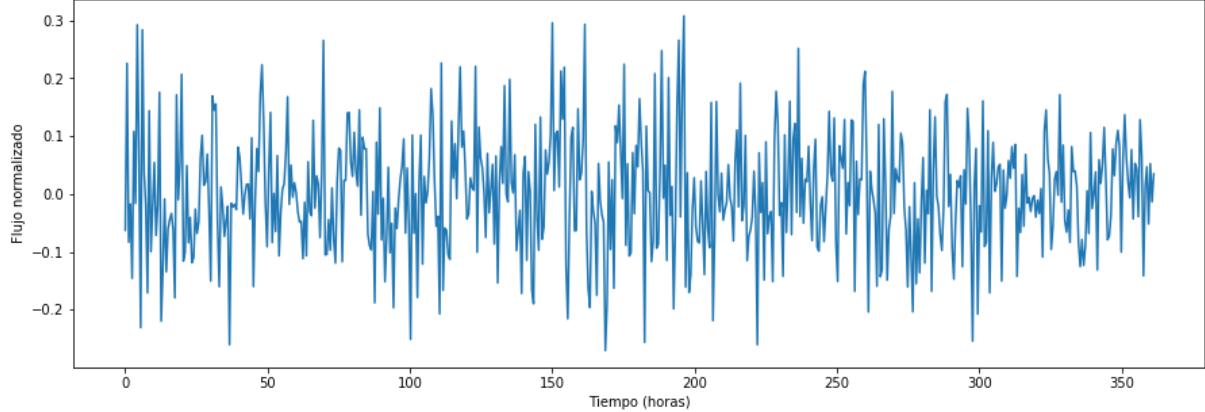
Flujo tras aplicar PCA de la estrella nº 16 CON exoplanetas



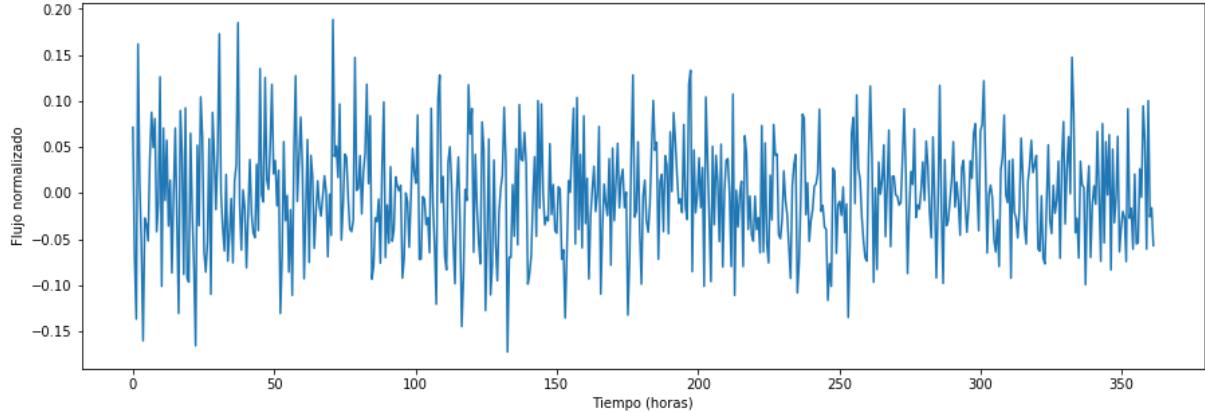
Flujo tras aplicar PCA de la estrella nº 17 CON exoplanetas



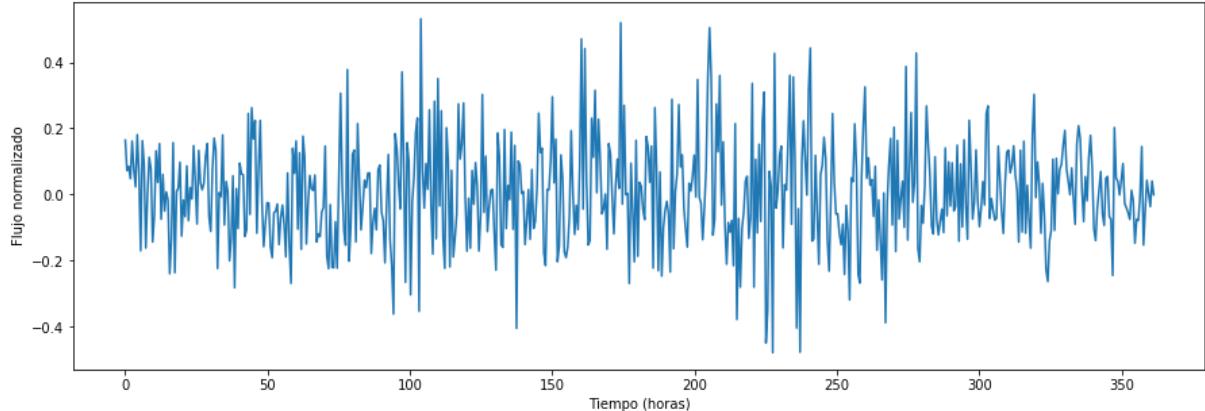
Flujo tras aplicar PCA de la estrella nº 18 CON exoplanetas



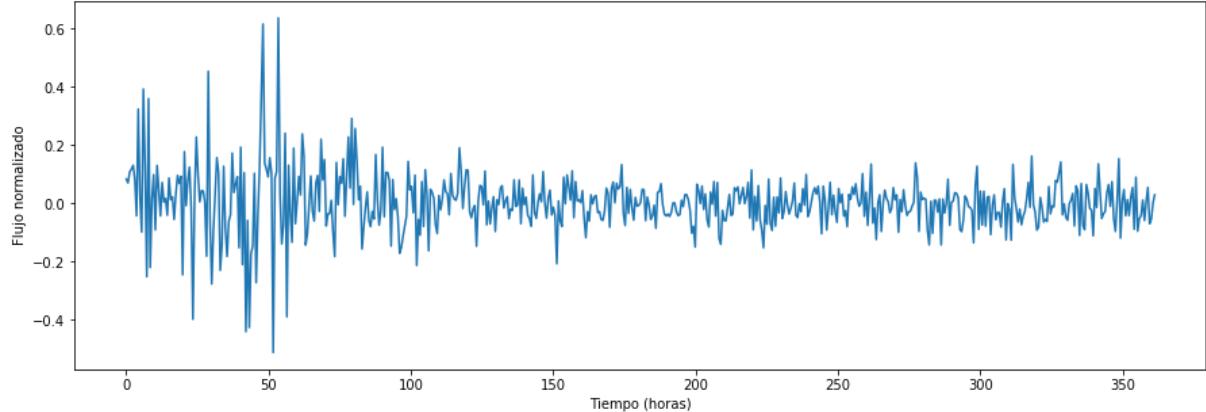
Flujo tras aplicar PCA de la estrella nº 19 CON exoplanetas



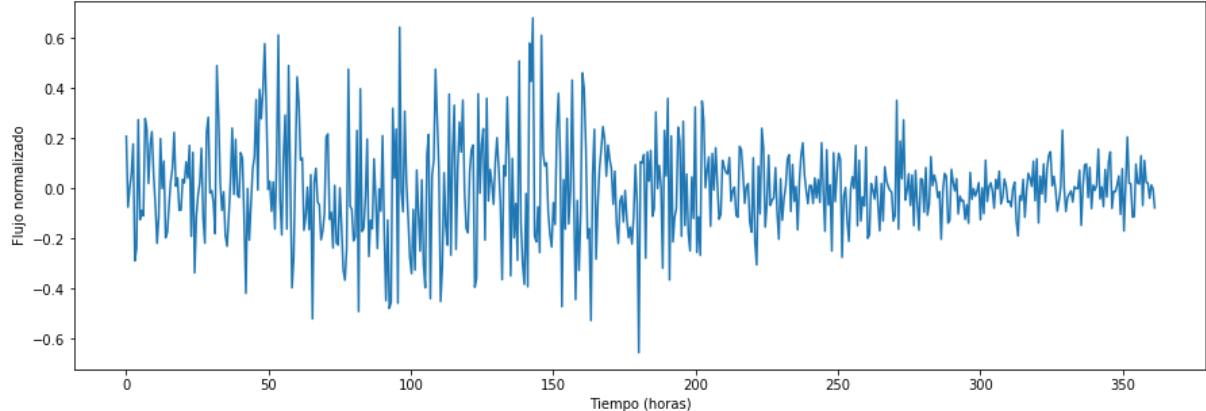
Flujo tras aplicar PCA de la estrella nº 20 CON exoplanetas



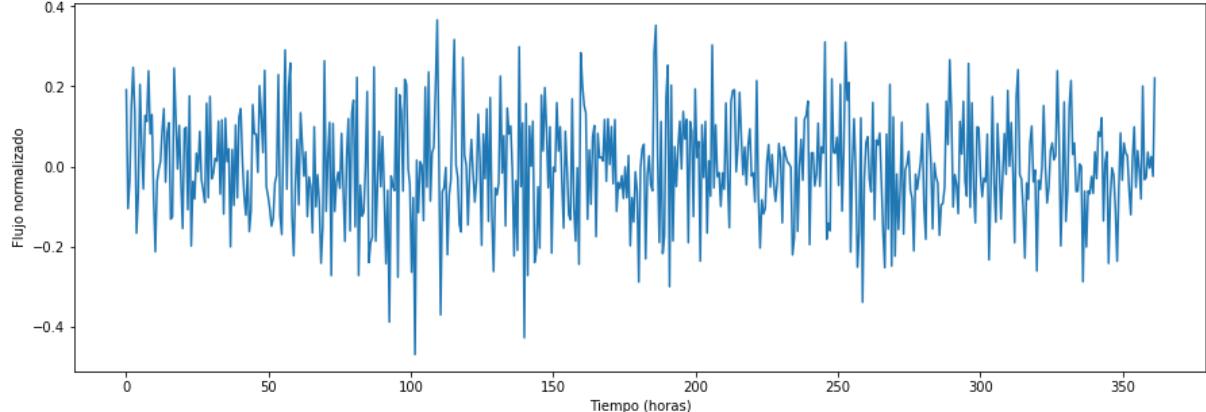
Flujo tras aplicar PCA de la estrella nº 21 CON exoplanetas



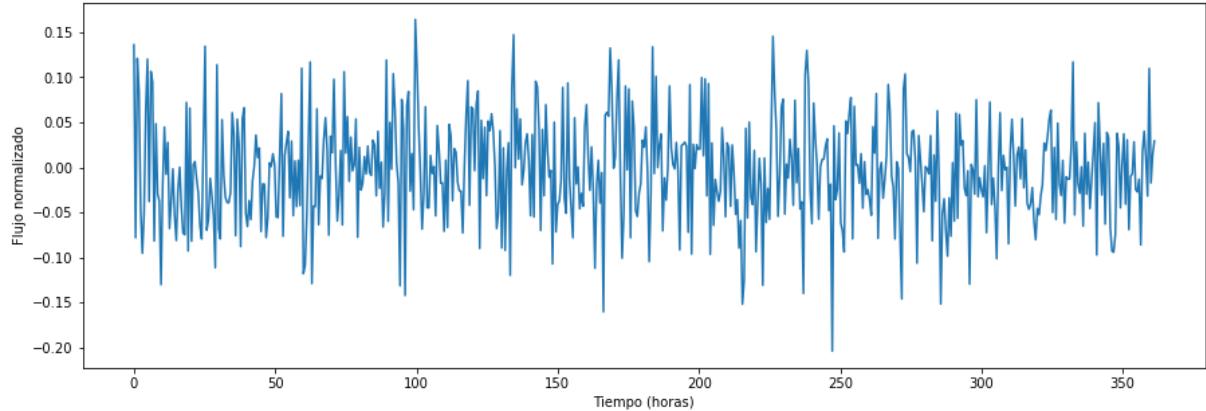
Flujo tras aplicar PCA de la estrella nº 22 CON exoplanetas



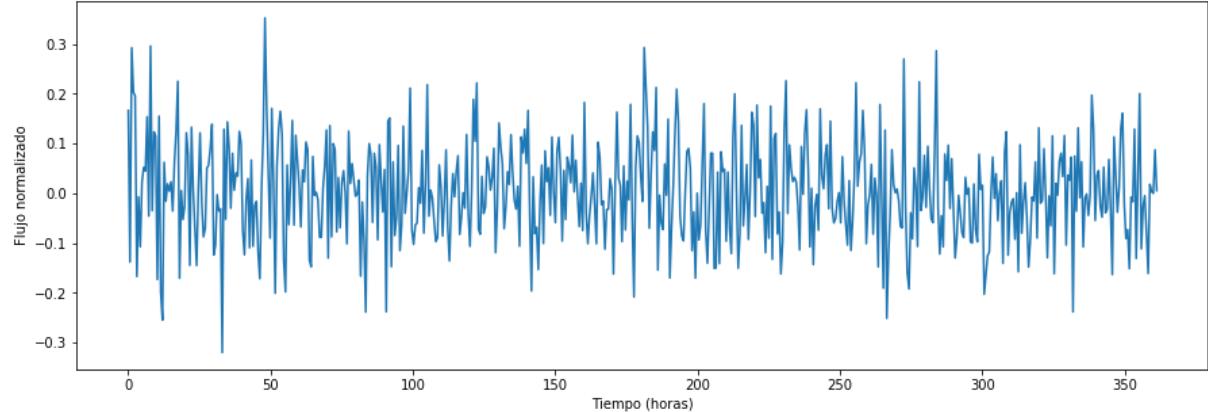
Flujo tras aplicar PCA de la estrella nº 23 CON exoplanetas



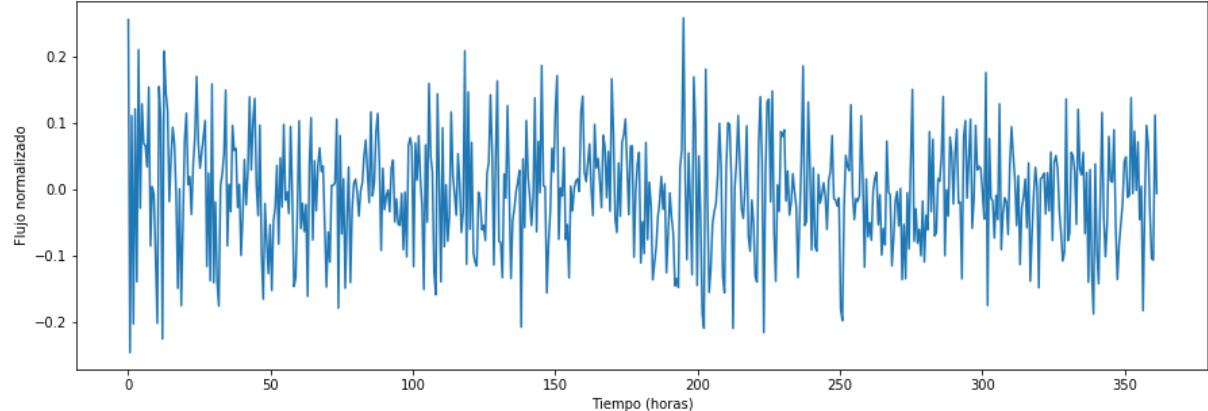
Flujo tras aplicar PCA de la estrella nº 24 CON exoplanetas



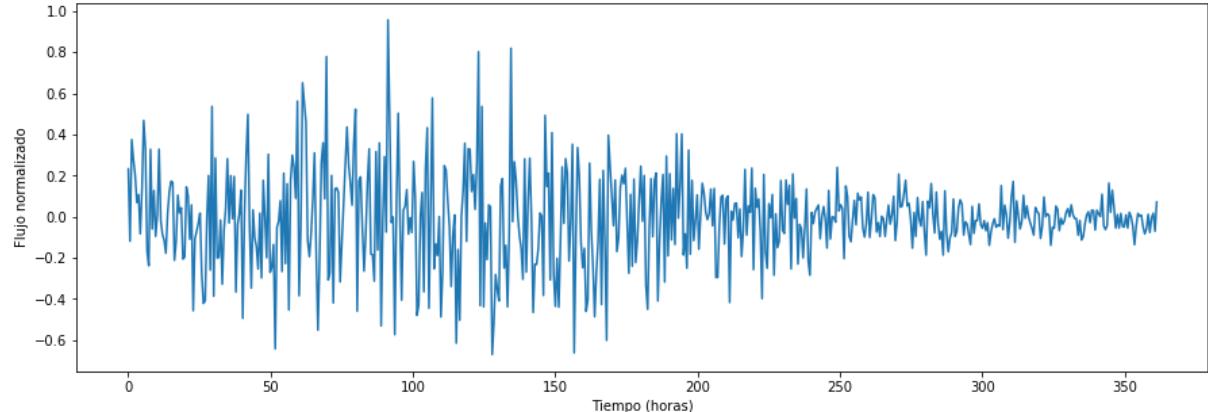
Flujo tras aplicar PCA de la estrella nº 25 CON exoplanetas



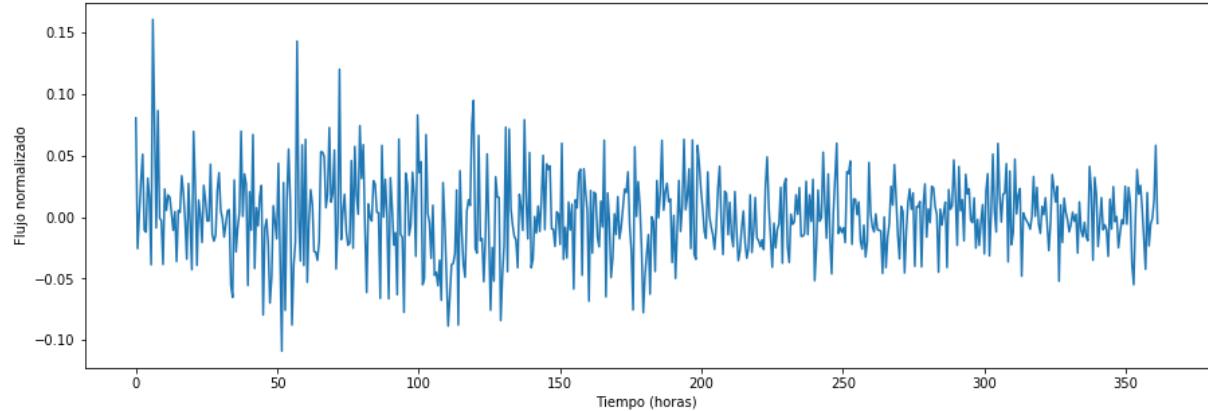
Flujo tras aplicar PCA de la estrella nº 26 CON exoplanetas



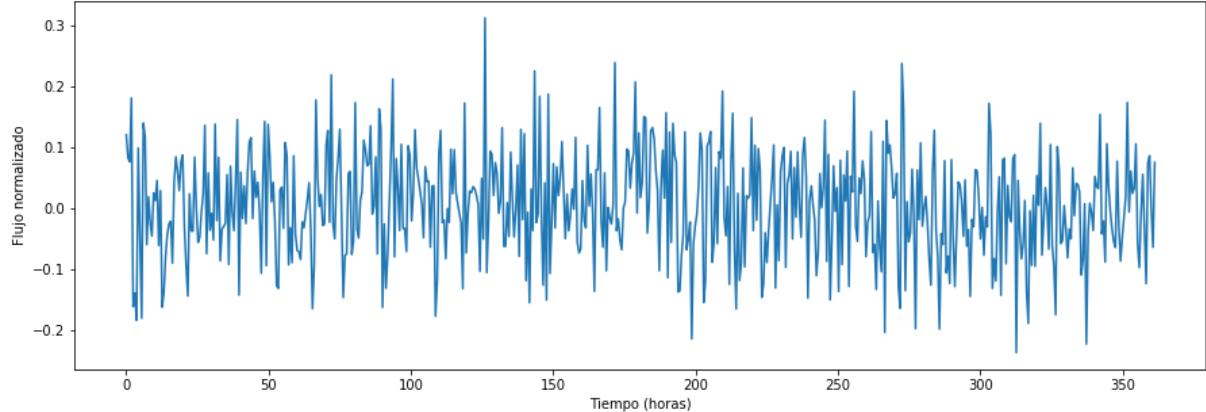
Flujo tras aplicar PCA de la estrella nº 27 CON exoplanetas



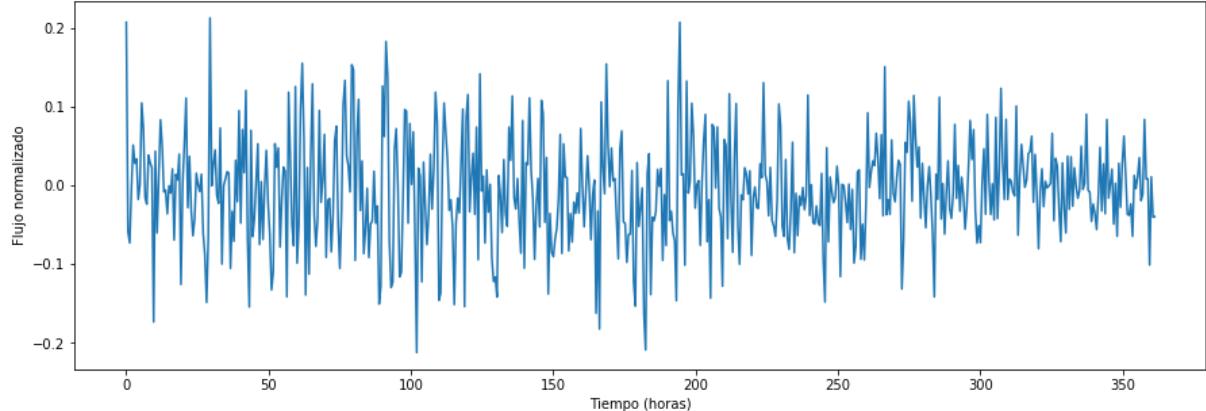
Flujo tras aplicar PCA de la estrella nº 28 CON exoplanetas



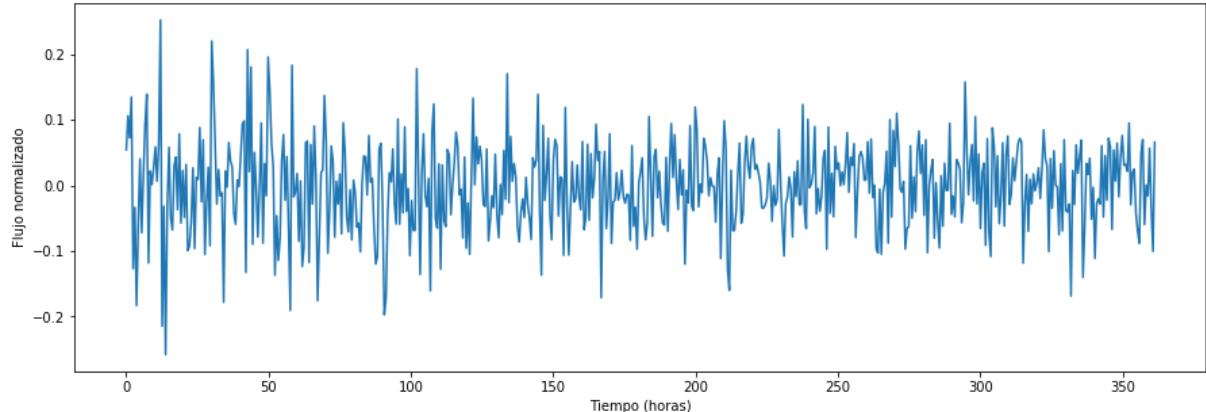
Flujo tras aplicar PCA de la estrella nº 29 CON exoplanetas



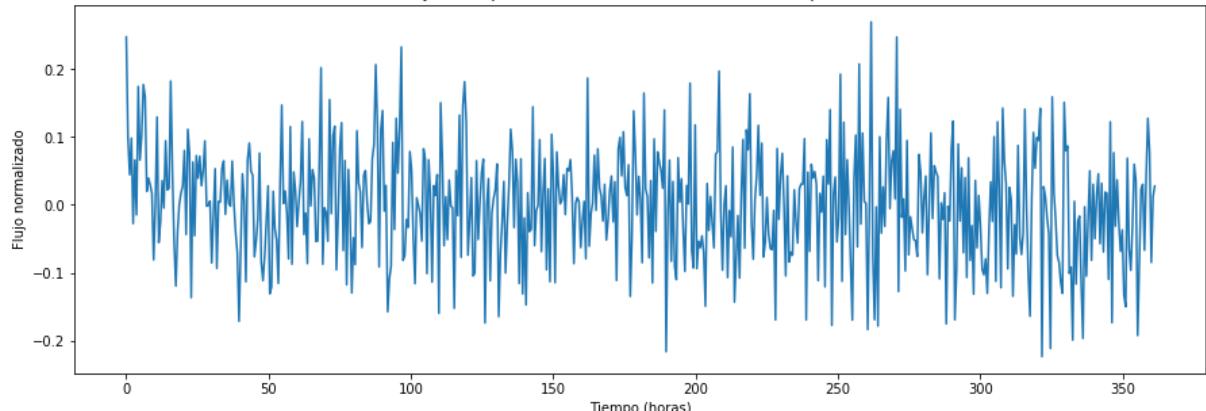
Flujo tras aplicar PCA de la estrella nº 30 CON exoplanetas



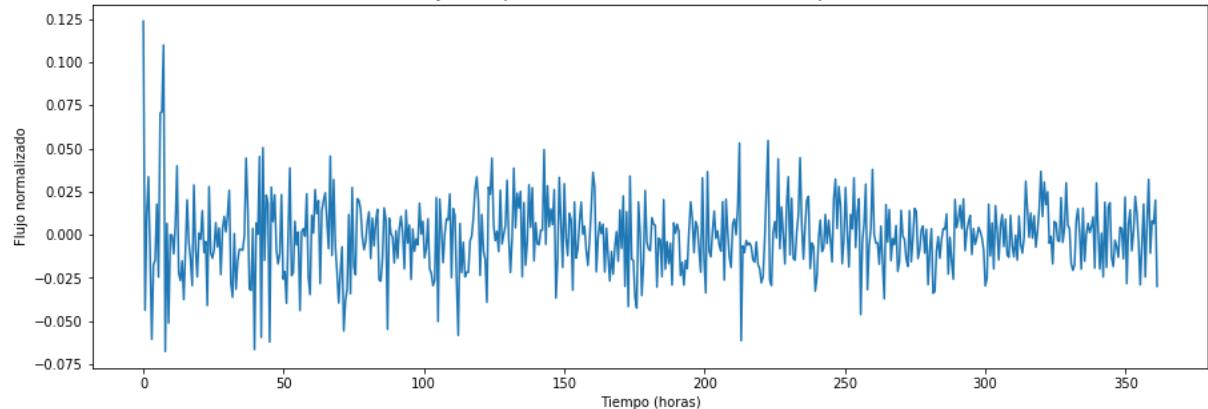
Flujo tras aplicar PCA de la estrella nº 31 CON exoplanetas



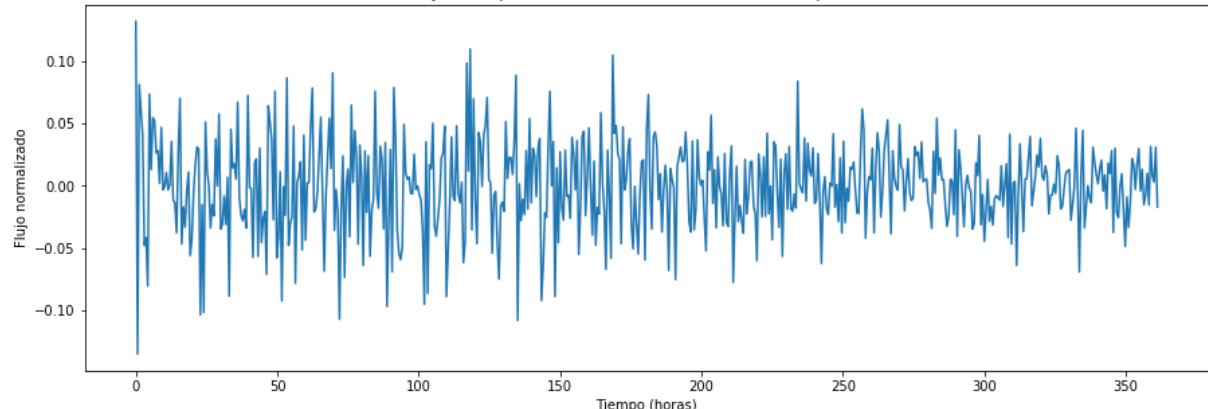
Flujo tras aplicar PCA de la estrella nº 32 CON exoplanetas



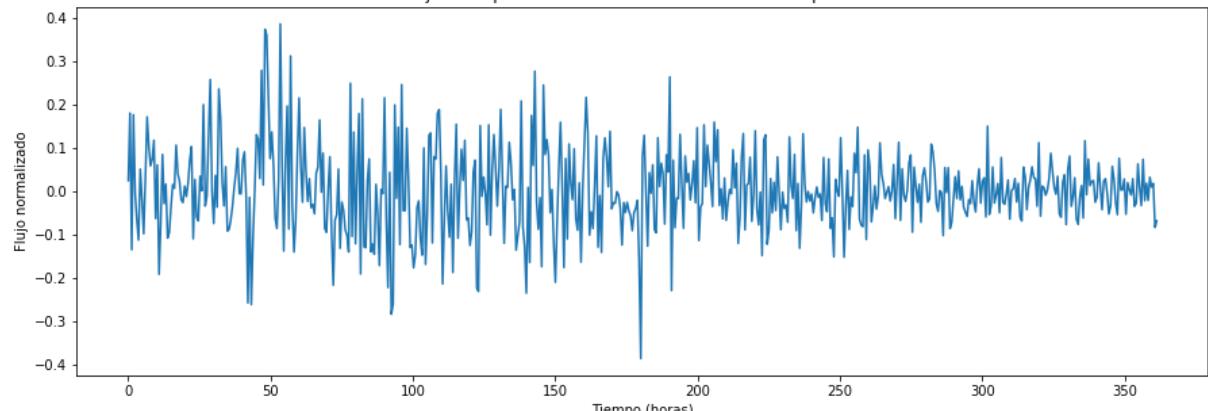
Flujo tras aplicar PCA de la estrella nº 33 CON exoplanetas



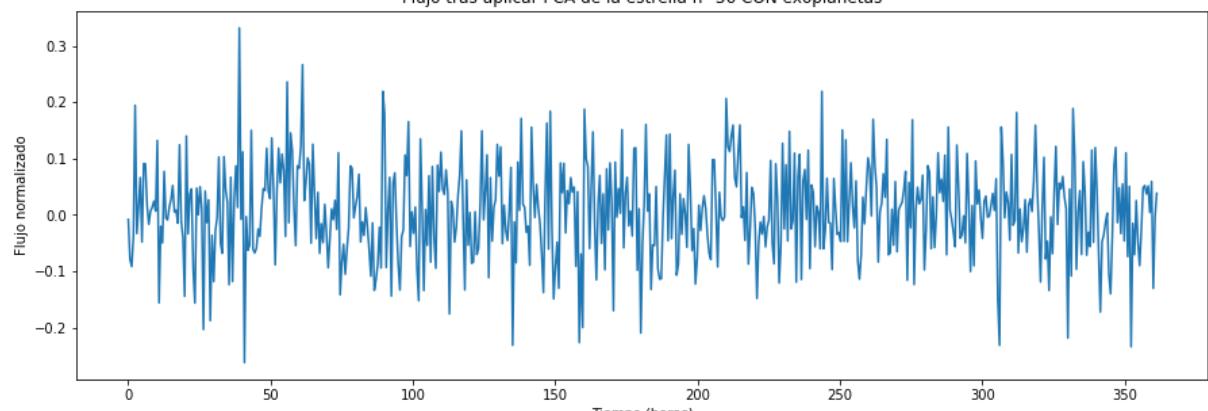
Flujo tras aplicar PCA de la estrella nº 34 CON exoplanetas

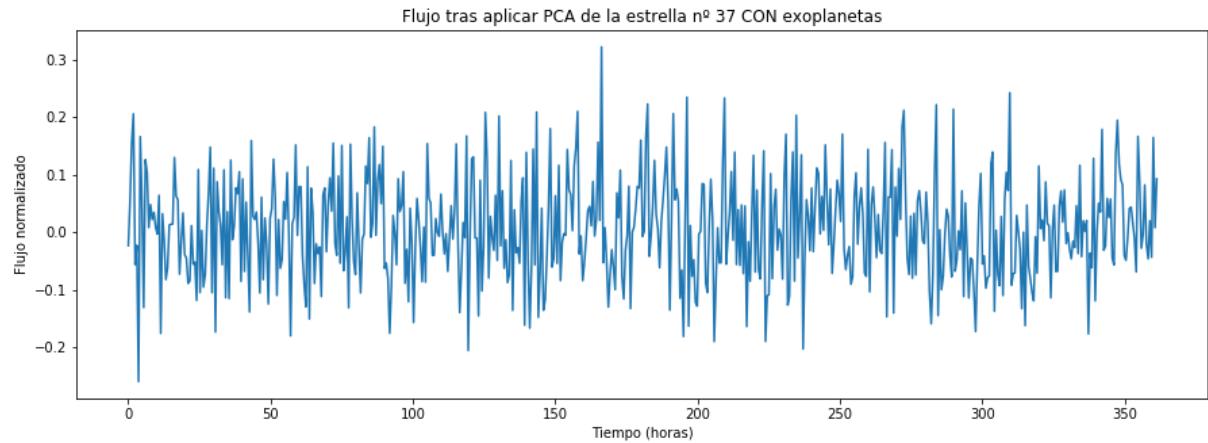


Flujo tras aplicar PCA de la estrella nº 35 CON exoplanetas



Flujo tras aplicar PCA de la estrella nº 36 CON exoplanetas





El objetivo es analizar cómo replicar estos datos y alterarlos de tal forma que no se pierda la información esencial que hace que sean detectados como exoplanetas.

Tras aplicar *PCA*, las dimensiones se encuentran ordenadas según su relevancia para el modelo, por lo que alterar su orden implicaría esa pérdida de información, además de que las curvas de luz se producen con ciertos patrones que, de verse alterados, modificarían por completo la naturaleza de los datos. Por lo tanto, alterar el orden de las columnas no se presenta como una opción.

Las dos mejores opciones, dados los datos, son:

- Incremento y decremento de una magnitud de los datos
- Aplicación de ruido a todos los datos

Incremento y decremento de magnitudes

Las magnitudes que se van a aplicar a los datos son: 0.02, 0.04, 0.06, 0.08 y 0.1.

De esta forma, los datos de estrellas con exoplanetas resultantes serán 407 en total

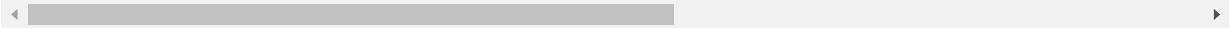
Estos son los datos que se van a replicar con los incrementos y decrementos

```
In [22]: flujos_exo = kepler_train_data_pca[kepler_train_data_pca.LABEL == 2].drop(['LABEL'], axis = 1) #ELiminuar la columna etiqueta y cargar los flujos de las estrellas con exoplanetas
flujos_exo
```

Out[22]:

	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	FLUX.10	...	FLUX.11
0	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	-0.350702	...	0.1561
1	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	-0.067823	...	-0.0491
2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	-0.959701	...	-0.0351
3	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	-0.506663	...	0.0271
4	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	-0.478106	...	-0.0851
5	0.061140	-0.240865	0.011431	-0.007067	-0.031689	-0.056541	-0.053235	0.088628	0.011557	-0.181824	...	-0.0341
6	0.197293	-0.160313	0.230235	0.015163	0.030869	-0.119705	0.058090	-0.110619	-0.133240	0.013793	...	0.0021
7	0.249868	-0.086585	-0.032370	0.087560	0.424559	-0.197987	0.067836	0.736586	0.180545	0.542623	...	-0.0381
8	0.088768	0.334573	-0.118508	0.191989	-0.095794	0.016415	-0.221840	0.068177	0.263669	-0.133538	...	0.0691
9	0.301405	0.555590	-0.223318	0.568477	0.088745	0.084723	-0.156697	0.518190	0.135297	-0.358419	...	0.0011
10	0.266852	-0.067526	0.214391	0.092154	0.110190	-0.041592	0.125091	-0.113500	0.066755	0.144230	...	-0.0501
11	0.284686	0.003130	0.267013	-0.039450	-0.145251	-0.081310	-0.097526	0.156078	-0.205338	-0.025936	...	-0.0141
12	0.089583	-0.266112	0.208647	-0.087560	-0.060132	-0.066580	0.030227	-0.089914	0.114521	0.091029	...	0.0821
13	0.133818	0.164683	-0.217678	0.066353	0.059936	0.257396	0.008733	-0.063664	0.079370	-0.520474	...	-0.0491
14	0.106546	-0.015737	0.015775	0.019291	-0.075444	-0.070158	-0.068381	-0.025959	0.019187	-0.004702	...	-0.0241
15	0.122226	0.259618	-0.381242	0.230716	-0.136526	0.142821	-0.457773	0.327960	-0.293475	-0.455511	...	0.1241
16	0.046697	0.301551	-0.281571	0.101372	-0.318933	0.144780	-0.383551	0.187950	0.111141	-0.320073	...	0.0591
17	-0.062804	0.225816	-0.083565	-0.017691	-0.145945	0.107940	-0.016525	0.292359	0.065911	-0.231102	...	-0.0391
18	0.071252	-0.068770	-0.136778	0.161818	0.010730	-0.054391	-0.160407	-0.027216	-0.034319	-0.051904	...	-0.0541
19	0.164294	0.072392	0.085188	0.048140	0.161650	0.073306	0.023376	0.181163	0.066656	-0.169957	...	-0.0781
20	0.083936	0.070635	0.108439	0.117677	0.131183	0.082125	-0.042198	0.324002	0.025082	-0.098155	...	-0.0511
21	0.208668	-0.073942	0.002424	0.059735	0.178019	-0.287884	-0.239169	0.274529	-0.123908	-0.087062	...	0.0181
22	0.191196	-0.105330	-0.035102	0.151725	0.247035	0.133942	-0.167036	-0.066051	0.204817	0.053060	...	-0.0071
23	0.135983	-0.078153	0.120900	0.085201	-0.047598	-0.095387	-0.051119	0.063816	0.120165	-0.037942	...	-0.0131
24	0.166286	-0.137955	0.292426	0.202830	0.195781	-0.167012	-0.006981	-0.107247	0.022148	0.051897	...	-0.1111
25	0.255150	-0.245629	0.110702	-0.202621	0.121012	-0.139176	0.209981	-0.028458	0.128523	0.068497	...	0.0041
26	0.231915	-0.118782	0.375190	0.277594	0.202669	0.069359	0.107793	-0.082906	0.134829	0.467953	...	0.0081
27	0.080522	-0.025662	0.001143	0.029867	0.051088	-0.010605	-0.012177	0.031581	0.015695	-0.038753	...	0.0071
28	0.120385	0.082462	0.076401	0.180972	-0.162037	-0.139504	-0.184744	0.098698	-0.047891	-0.181057	...	-0.0981
29	0.207366	-0.058673	-0.072995	-0.004095	0.051369	0.028015	0.033668	-0.017668	0.001356	0.104920	...	0.0351
30	0.054117	0.106207	0.072326	0.134578	-0.126938	-0.033638	-0.183282	-0.059246	0.040402	-0.072322	...	-0.0881
31	0.247725	0.098341	0.044400	0.098426	-0.027612	0.066300	-0.014687	0.174721	0.066176	0.103500	...	-0.0811
32	0.123771	-0.043831	0.011647	0.033579	-0.011514	-0.060634	-0.017527	-0.014784	0.017666	-0.024588	...	-0.0061
33	0.132017	-0.135424	0.081418	0.060945	0.040443	-0.047977	-0.041756	-0.080637	0.073673	0.012950	...	0.0131
34	0.024471	0.180167	-0.135157	0.176437	-0.004657	-0.066792	-0.112622	0.050700	-0.017005	-0.098211	...	-0.0331
35	-0.008267	-0.079213	-0.091846	-0.037482	0.194317	-0.033444	0.013973	0.065864	-0.048314	0.090901	...	-0.0271
36	-0.023589	0.051946	0.155949	0.205879	-0.056546	-0.022957	-0.260034	0.166154	0.053266	-0.131379	...	-0.0281

37 rows × 603 columns

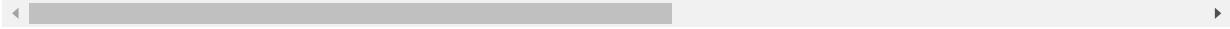
**Datos finales con los incrementos y decrementos**

```
In [23]: flujos_exo_incrdecr = pd.DataFrame(flujos_exo) #Crear nuevo DataFrame para todos los datos resultantes de este metodo
#Incrementos
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo + 0.02), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo + 0.04), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo + 0.06), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo + 0.08), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo + 0.1), ignore_index = True)
#Decremento
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo - 0.02), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo - 0.04), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo - 0.06), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo - 0.08), ignore_index = True)
flujos_exo_incrdecr = flujos_exo_incrdecr.append((flujos_exo - 0.1), ignore_index = True)
flujos_exo_incrdecr
```

Out[23]:

	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	FLUX.10	...	FLUX
0	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	-0.350702	...	0.150
1	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	-0.067823	...	-0.040
2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	-0.959701	...	-0.038
3	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	-0.506663	...	0.021
4	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	-0.478106	...	-0.081
...
402	0.023771	-0.143831	-0.088353	-0.066421	-0.111514	-0.160634	-0.117527	-0.114784	-0.082334	-0.124588	...	-0.101
403	0.032017	-0.235424	-0.018582	-0.039055	-0.059557	-0.147977	-0.141756	-0.180637	-0.026327	-0.087050	...	-0.081
404	-0.075529	0.080167	-0.235157	0.076437	-0.104657	-0.166792	-0.212622	-0.049300	-0.117005	-0.198211	...	-0.131
405	-0.108267	-0.179213	-0.191846	-0.137482	0.094317	-0.133444	-0.086027	-0.034136	-0.148314	-0.009099	...	-0.121
406	-0.123589	-0.048054	0.055949	0.105879	-0.156546	-0.122957	-0.360034	0.066154	-0.046734	-0.231379	...	-0.121

407 rows × 603 columns



Aplicación de ruido

A las 407 filas obtenidas se les van a aplicar 6 instancias de ruido con las varianzas siguientes, repectivamente: 0.000015, 0.00003, 0.000045, 0.00006, 0.00075 y 0.00009.

Este proceso dejará un total de 2849 filas equivalentes a estrellas con exoplanetas, todas ellas diferentes entre sí.

```
In [24]: #Generar Las 6 instancias de ruido con las varianzas correspondientes
ruido1 = np.random.normal(0, 0.000015, [407, 603])
ruido2 = np.random.normal(0, 0.00003, [407, 603])
ruido3 = np.random.normal(0, 0.000045, [407, 603])
ruido4 = np.random.normal(0, 0.00006, [407, 603])
ruido5 = np.random.normal(0, 0.000075, [407, 603])
ruido6 = np.random.normal(0, 0.00009, [407, 603])

#Añadir las instancias de ruido a los datos
flujos_exo_ruido1 = fluxos_exo_incrdecr + ruido1
flujos_exo_ruido2 = fluxos_exo_incrdecr + ruido2
flujos_exo_ruido3 = fluxos_exo_incrdecr + ruido3
flujos_exo_ruido4 = fluxos_exo_incrdecr + ruido4
flujos_exo_ruido5 = fluxos_exo_incrdecr + ruido5
flujos_exo_ruido6 = fluxos_exo_incrdecr + ruido6

#Crear nuevo DataFrame para todos los datos resultantes de este metodo
fluxos_exo_da = pd.DataFrame(fluxos_exo_incrdecr)

#Añadir todas las instancias finales en un unico DataFrame
fluxos_exo_da = fluxos_exo_da.append(fluxos_exo_ruido1, ignore_index = True)
fluxos_exo_da = fluxos_exo_da.append(fluxos_exo_ruido2, ignore_index = True)
fluxos_exo_da = fluxos_exo_da.append(fluxos_exo_ruido3, ignore_index = True)
fluxos_exo_da = fluxos_exo_da.append(fluxos_exo_ruido4, ignore_index = True)
fluxos_exo_da = fluxos_exo_da.append(fluxos_exo_ruido5, ignore_index = True)
fluxos_exo_da = fluxos_exo_da.append(fluxos_exo_ruido6, ignore_index = True)
fluxos_exo_da
```

Out[24]:

	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	FLUX.10	...	FLU
0	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	-0.350702	...	0.1:
1	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	-0.067823	...	-0.0:
2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	-0.959701	...	-0.0:
3	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	-0.506663	...	0.0:
4	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	-0.478106	...	-0.0:
...
2844	0.023684	-0.143729	-0.088381	-0.066401	-0.111587	-0.160832	-0.117576	-0.114774	-0.082425	-0.124735	...	-0.1:
2845	0.032063	-0.235370	-0.018679	-0.038811	-0.059494	-0.147944	-0.141696	-0.180650	-0.026279	-0.087008	...	-0.0:
2846	-0.075598	0.080178	-0.234849	0.076402	-0.104667	-0.166866	-0.212455	-0.049279	-0.116961	-0.198244	...	-0.1:
2847	-0.108298	-0.179033	-0.191851	-0.137490	0.094292	-0.133475	-0.086013	-0.034150	-0.148294	-0.008885	...	-0.1:
2848	-0.123608	-0.048203	0.055908	0.105944	-0.156517	-0.122955	-0.360031	0.065988	-0.046692	-0.231375	...	-0.1:

2849 rows × 603 columns

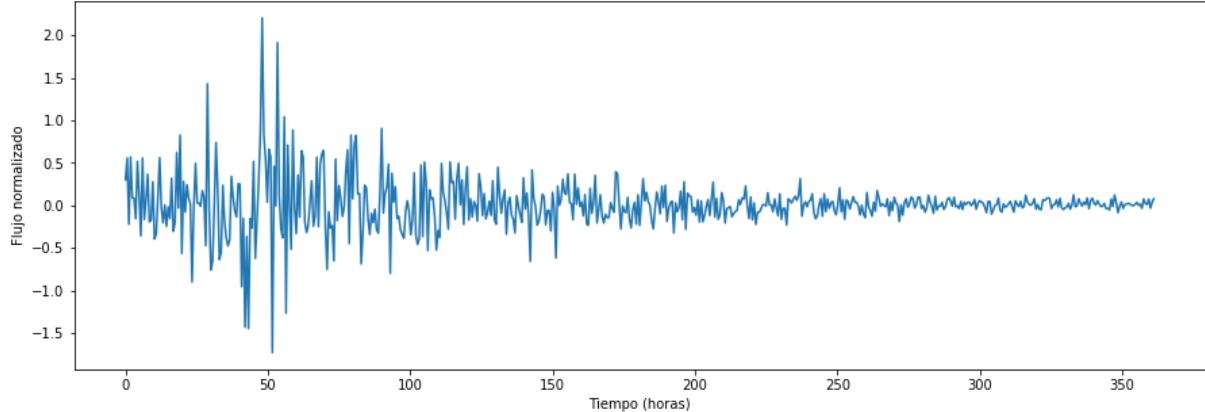
Arriba se muestran todos los datos aumentados, junto a los originales, de las 37 estrellas con exoplanetas iniciales.

Abajo se muestran los flujos de la estrella 10 con exoplanetas y algunas de sus modificaciones -ruido y/o incremento/decremento-. Como se puede apreciar, a simple vista no son muy diferentes entre sí, lo que es un buen indicativo de que la relatividad de los datos sigue presente pese a las modificaciones. Pero, por otro lado, los datos son lo suficientemente diferentes para entrenar el modelo evitando sobreajustes.

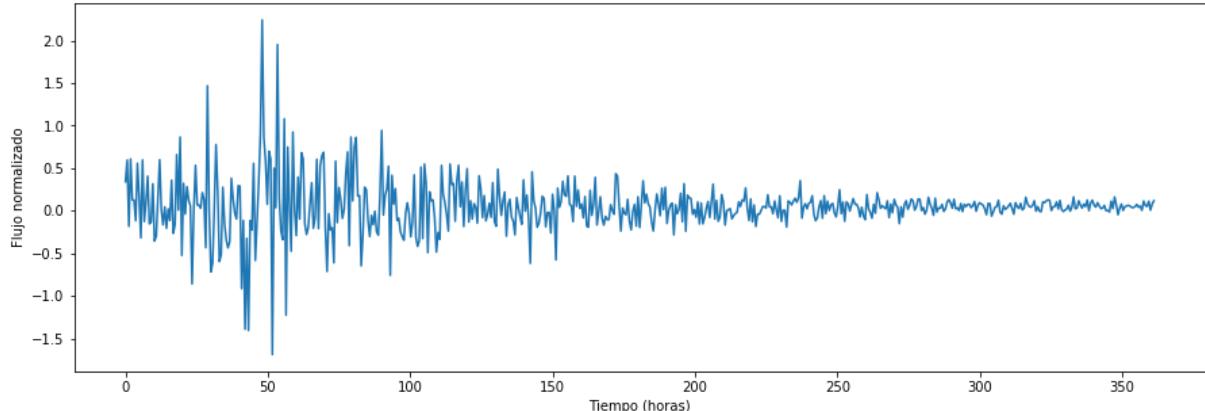
```
In [25]: i = 9
while i < (len(flujos_exo_da)-1):
    #Se extraen los datos de las distintas mediciones de luz, eliminando la columna "LABEL"
    flujo = flujos_exo_da.iloc[i,:]
    tiempo = np.arange(len(flujo)) * (36/60) #Variable "tiempo" en horas
    plt.figure(figsize=(15, 5)) #Tamaño del gráfico
    plt.title('Flujo de la estrella nº 9 CON exoplanetas')
    plt.xlabel('Tiempo (horas)')
    plt.ylabel('Flujo normalizado')
    plt.plot(tiempo, flujo)
    plt.rcParams.update({'figure.max_open_warning': 0})
    i += 74

plt.show()
```

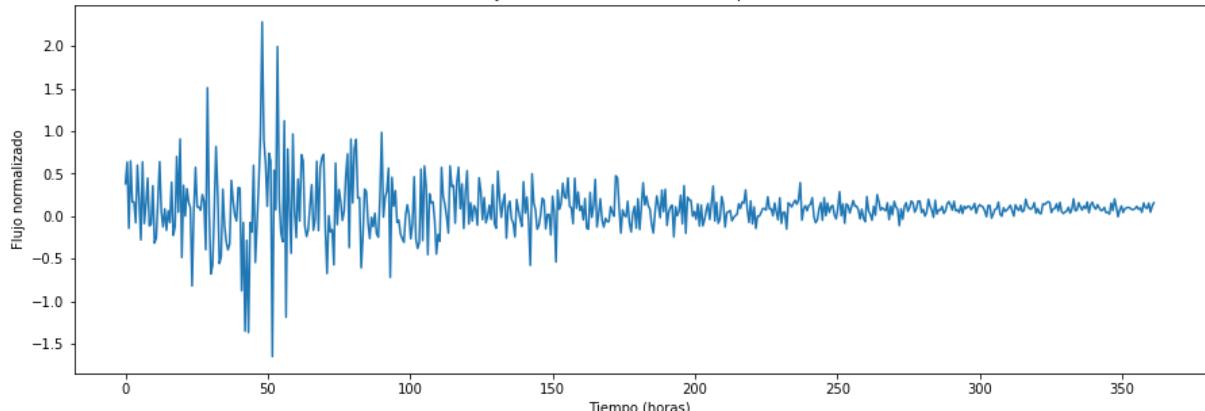
Flujo de la estrella nº 9 CON exoplanetas



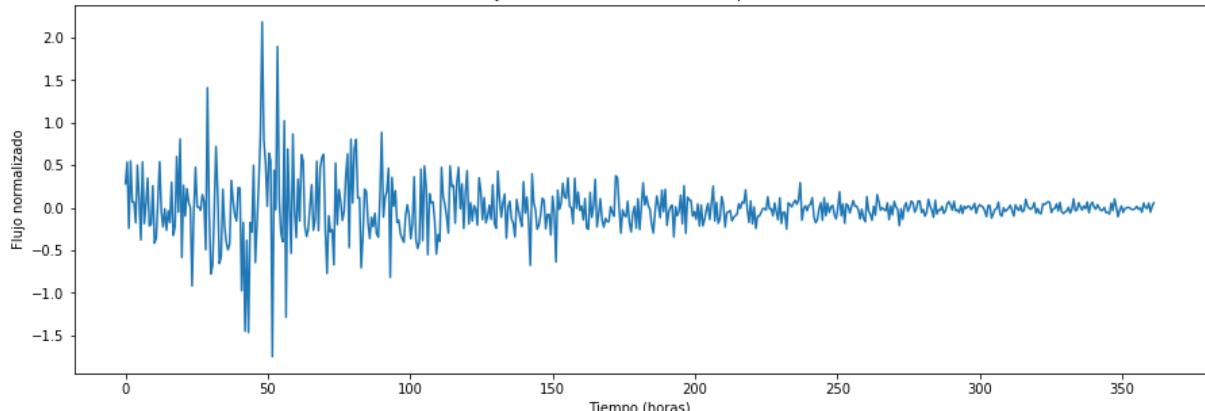
Flujo de la estrella nº 9 CON exoplanetas



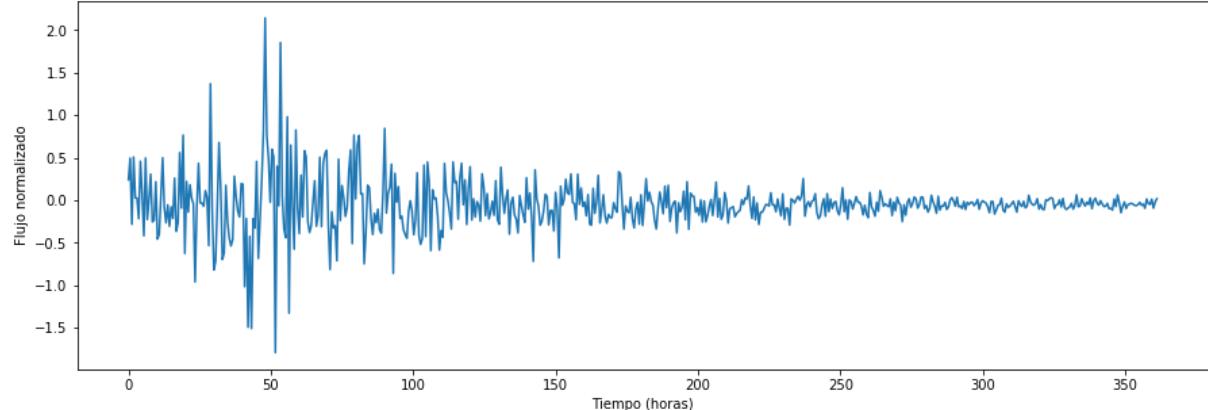
Flujo de la estrella nº 9 CON exoplanetas



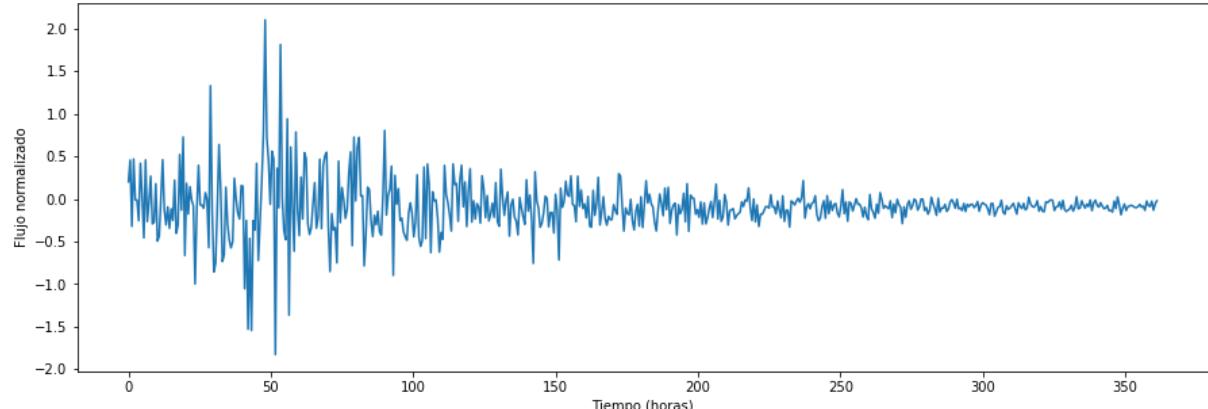
Flujo de la estrella nº 9 CON exoplanetas



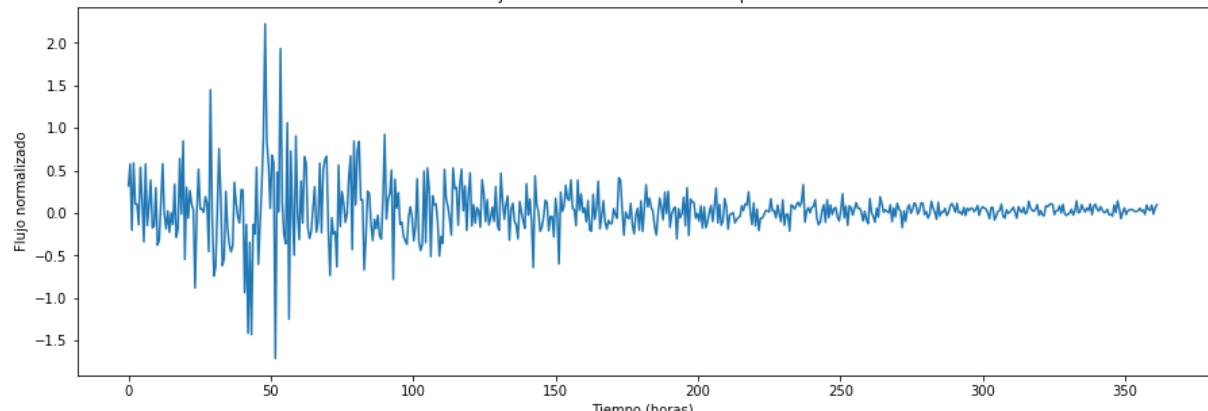
Flujo de la estrella nº 9 CON exoplanetas



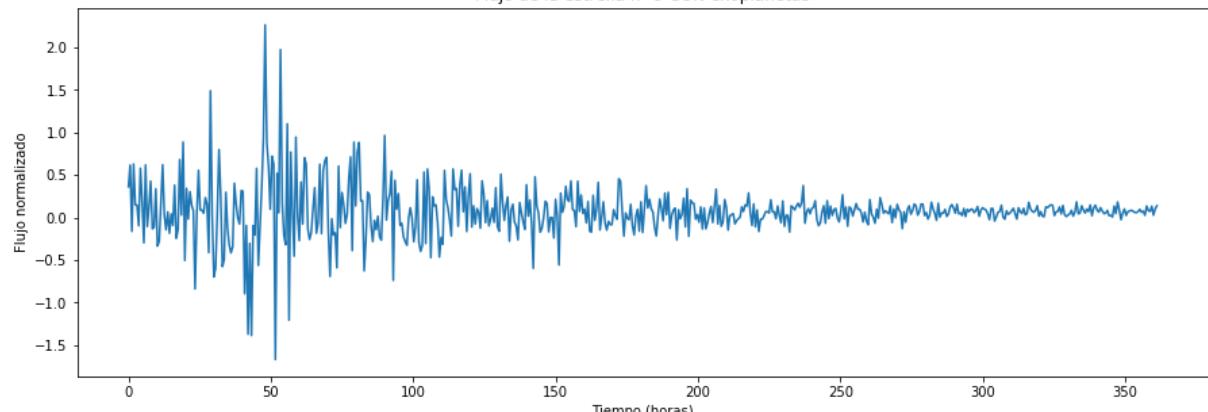
Flujo de la estrella nº 9 CON exoplanetas

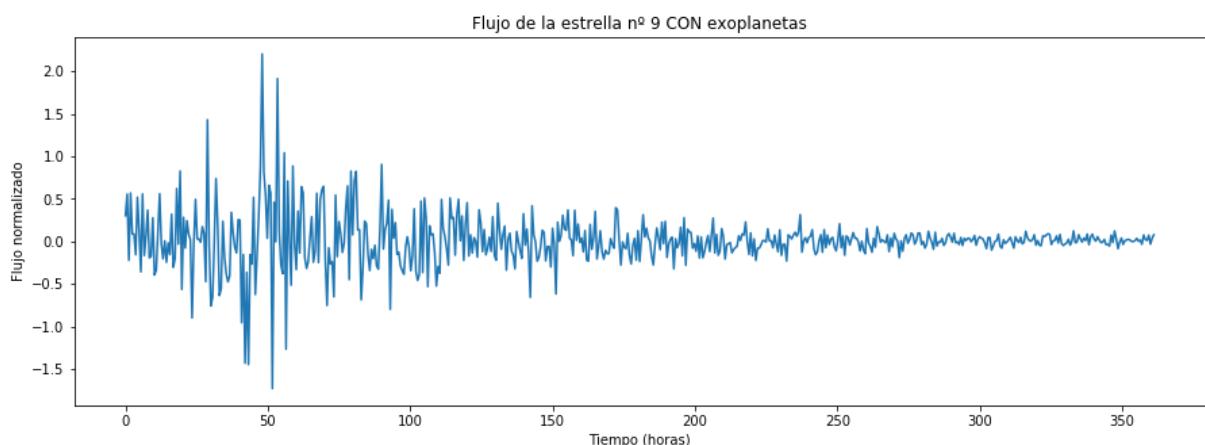
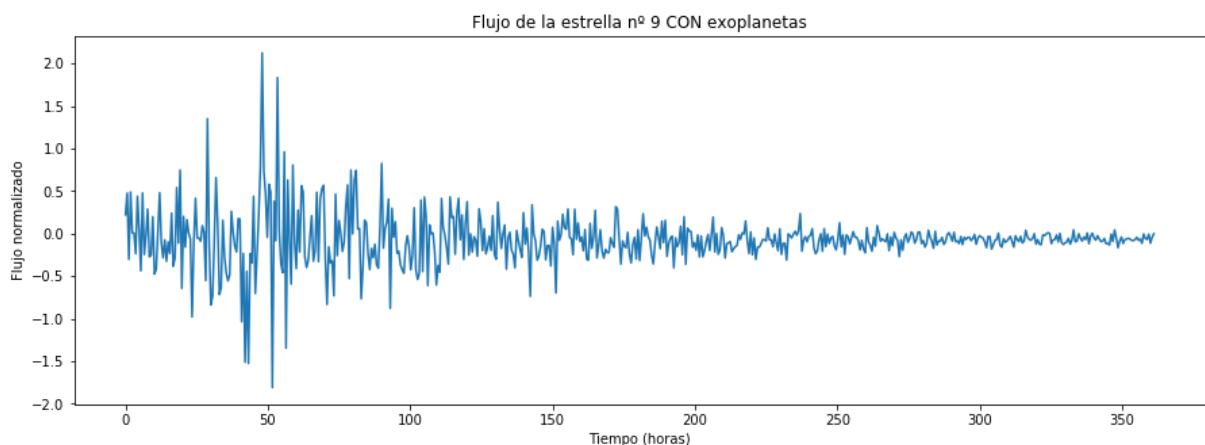
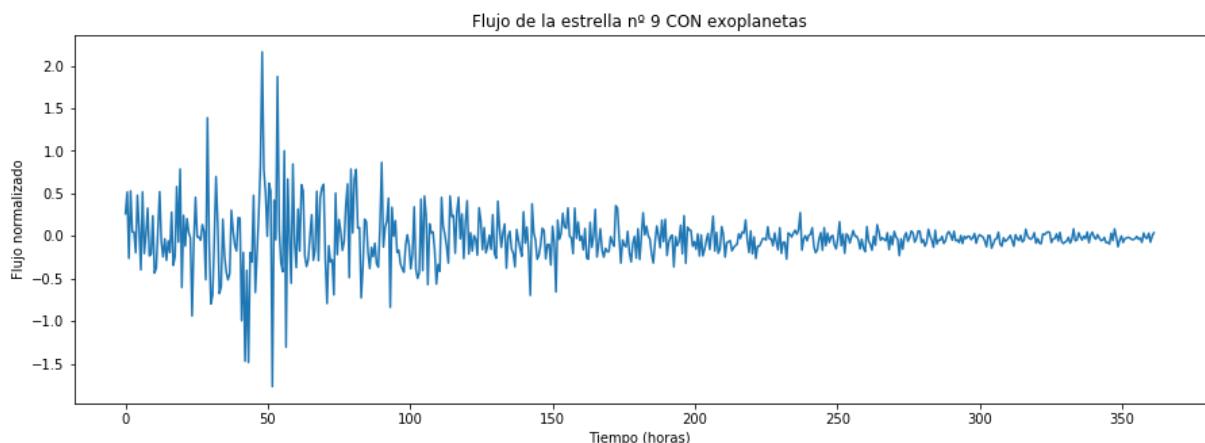
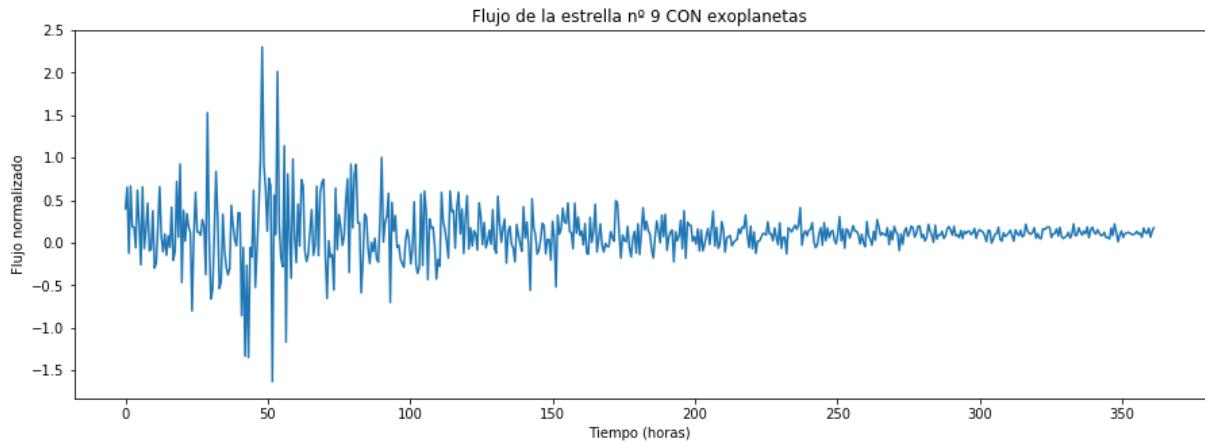


Flujo de la estrella nº 9 CON exoplanetas

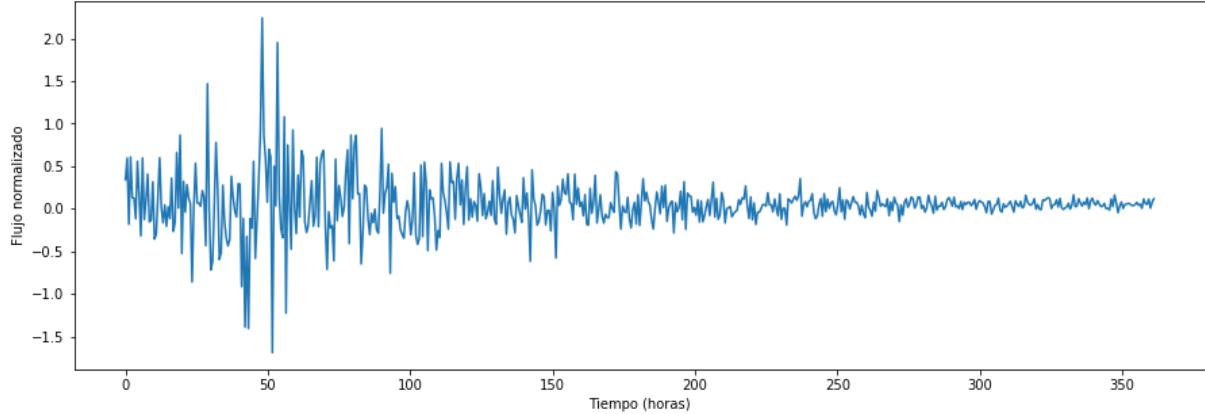


Flujo de la estrella nº 9 CON exoplanetas

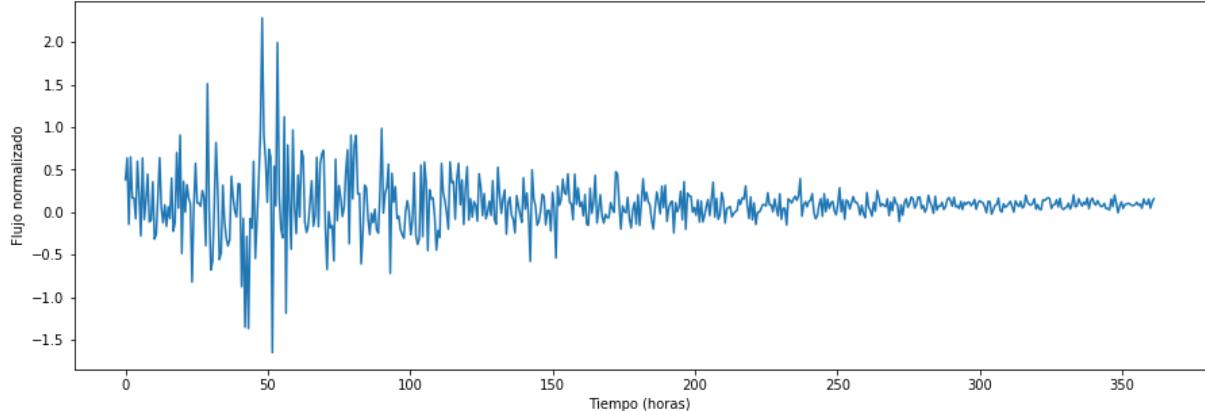




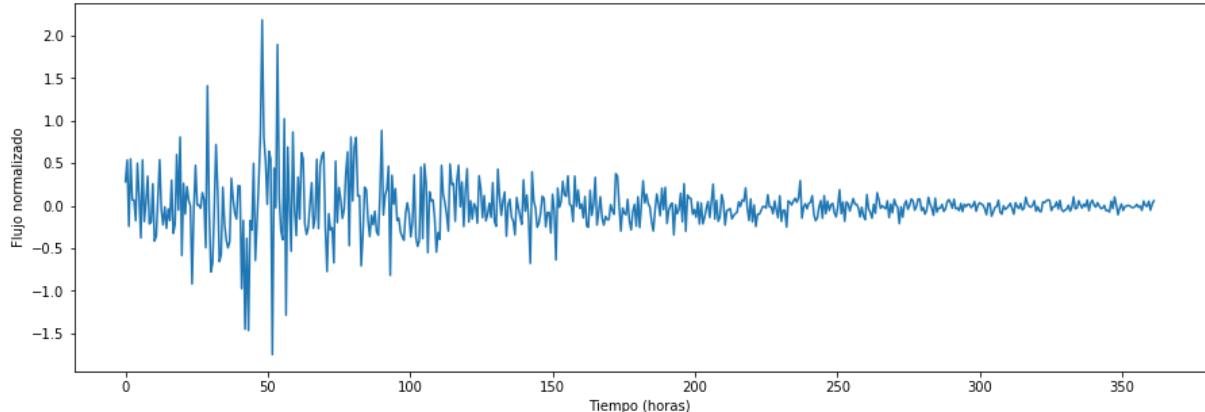
Flujo de la estrella nº 9 CON exoplanetas



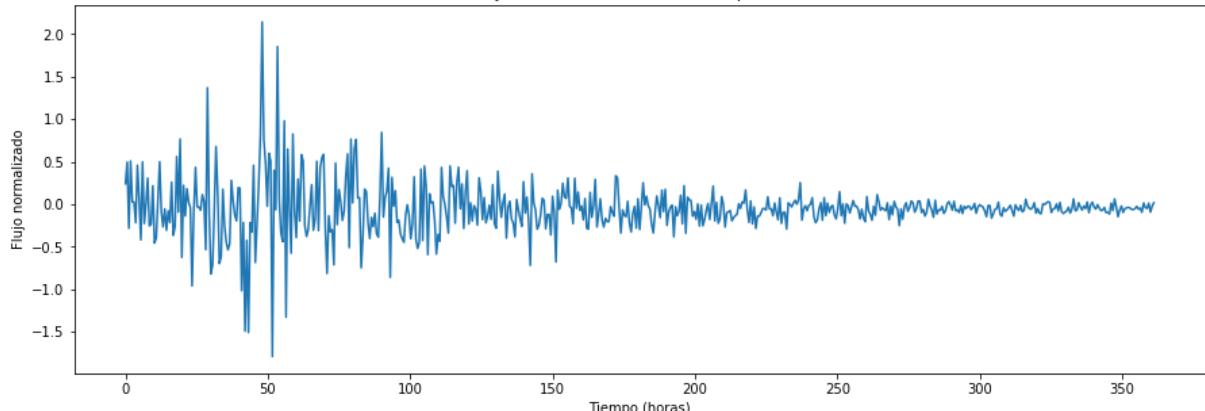
Flujo de la estrella nº 9 CON exoplanetas



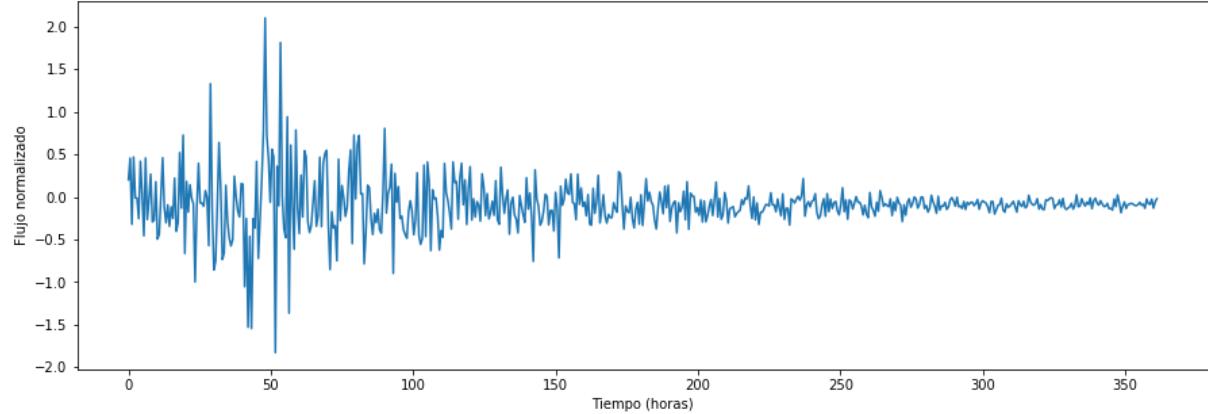
Flujo de la estrella nº 9 CON exoplanetas



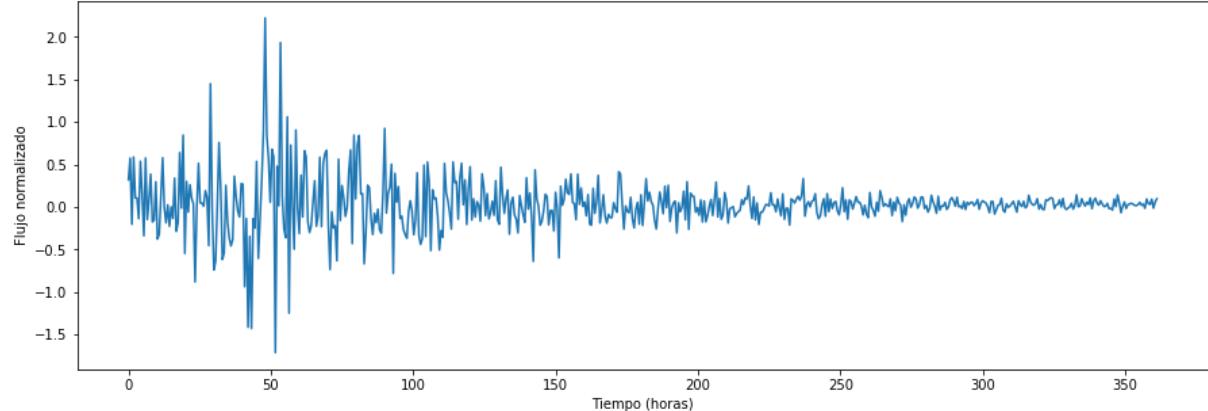
Flujo de la estrella nº 9 CON exoplanetas



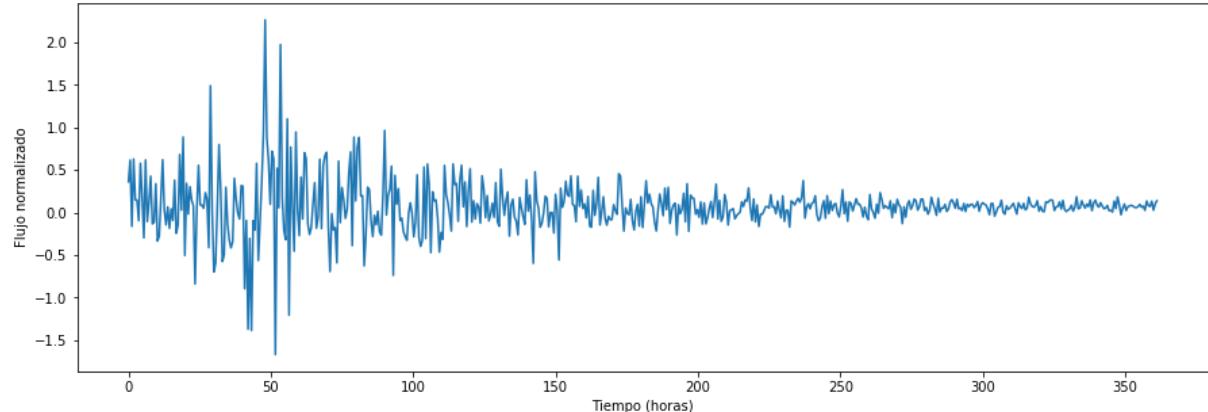
Flujo de la estrella nº 9 CON exoplanetas



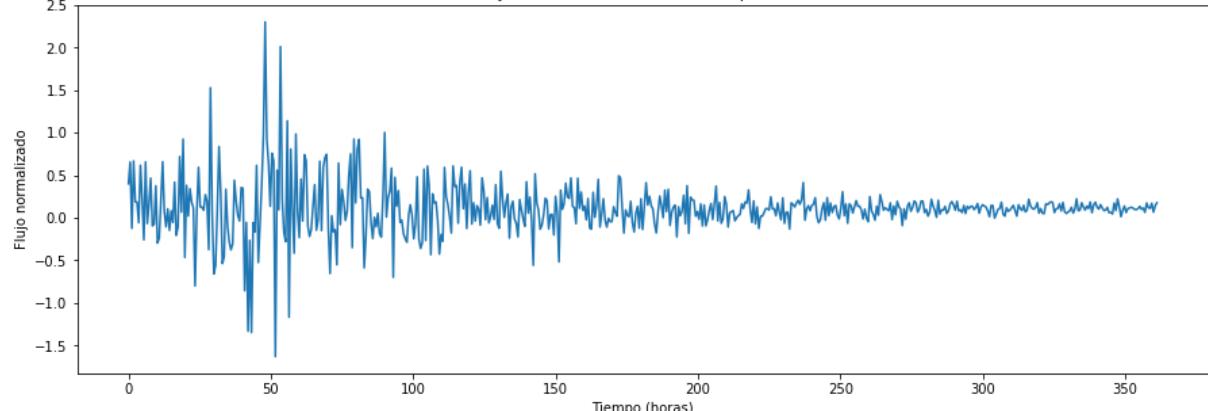
Flujo de la estrella nº 9 CON exoplanetas



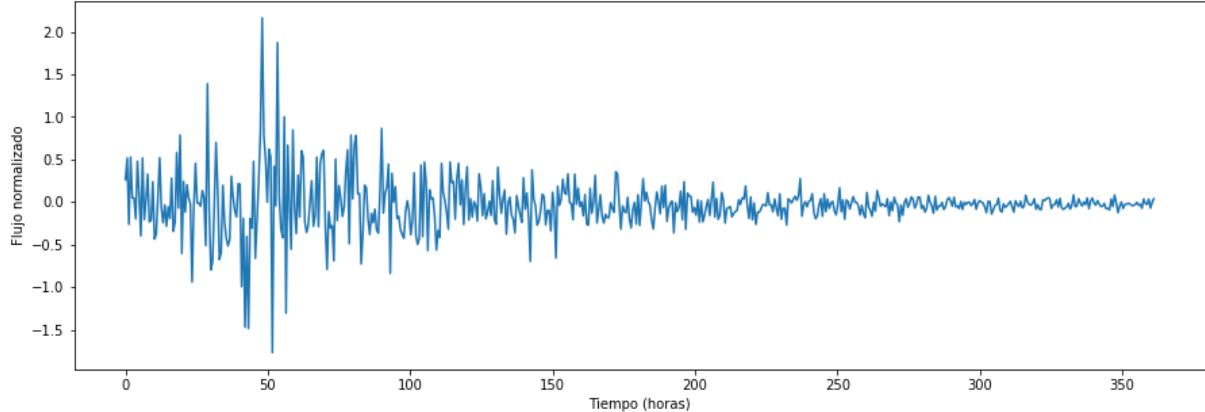
Flujo de la estrella nº 9 CON exoplanetas



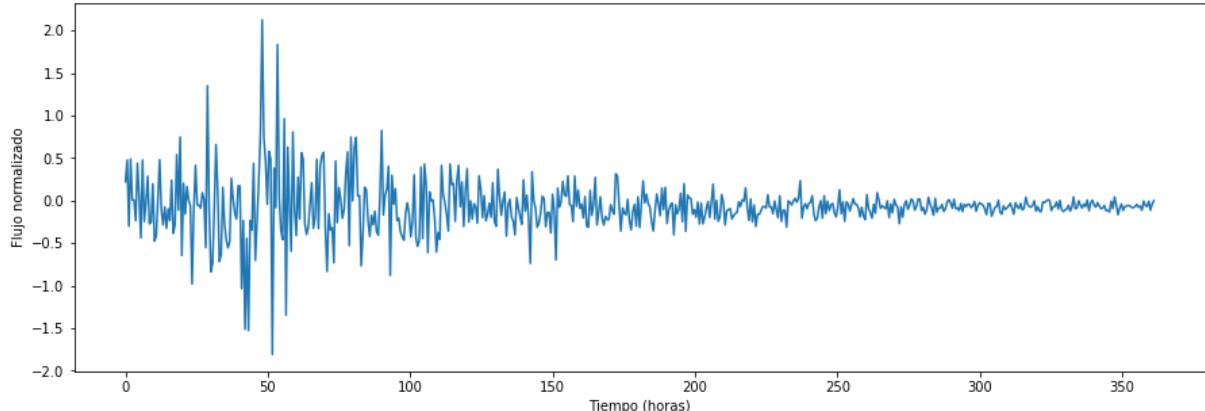
Flujo de la estrella nº 9 CON exoplanetas



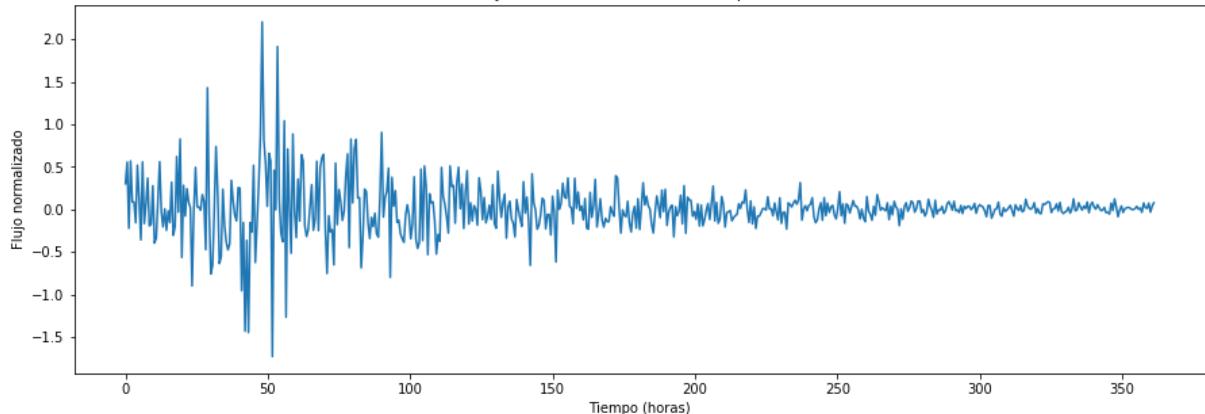
Flujo de la estrella nº 9 CON exoplanetas



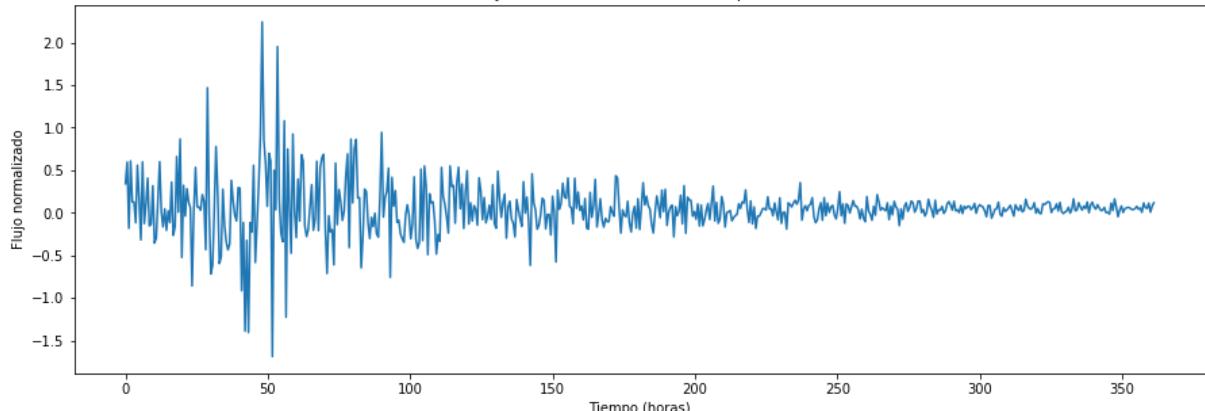
Flujo de la estrella nº 9 CON exoplanetas



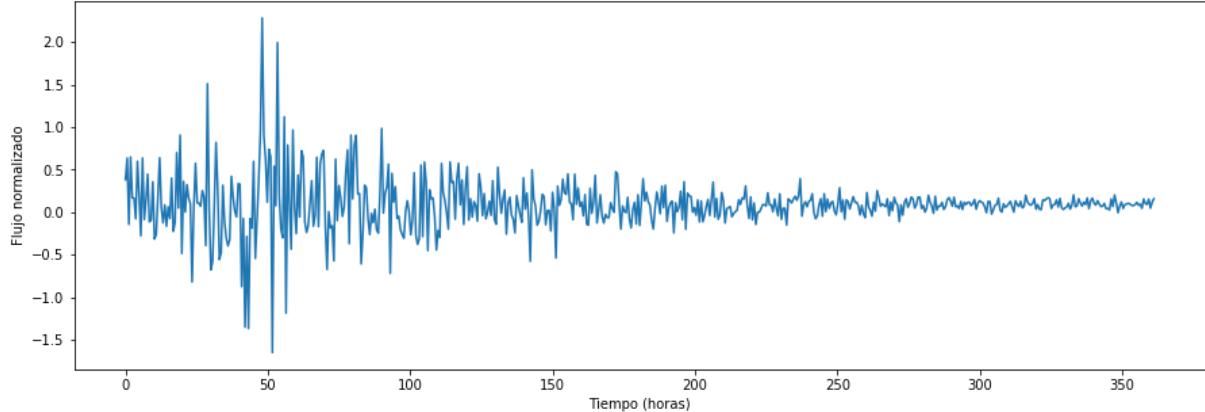
Flujo de la estrella nº 9 CON exoplanetas



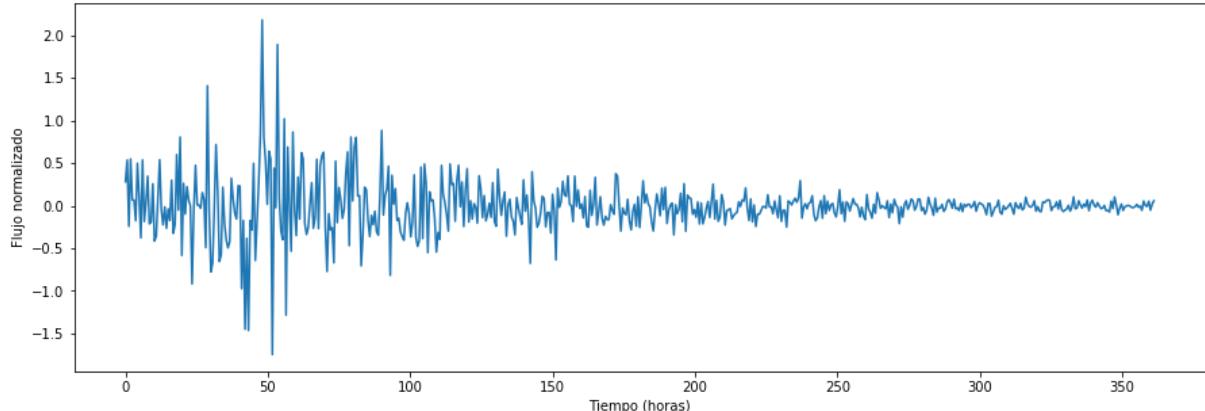
Flujo de la estrella nº 9 CON exoplanetas



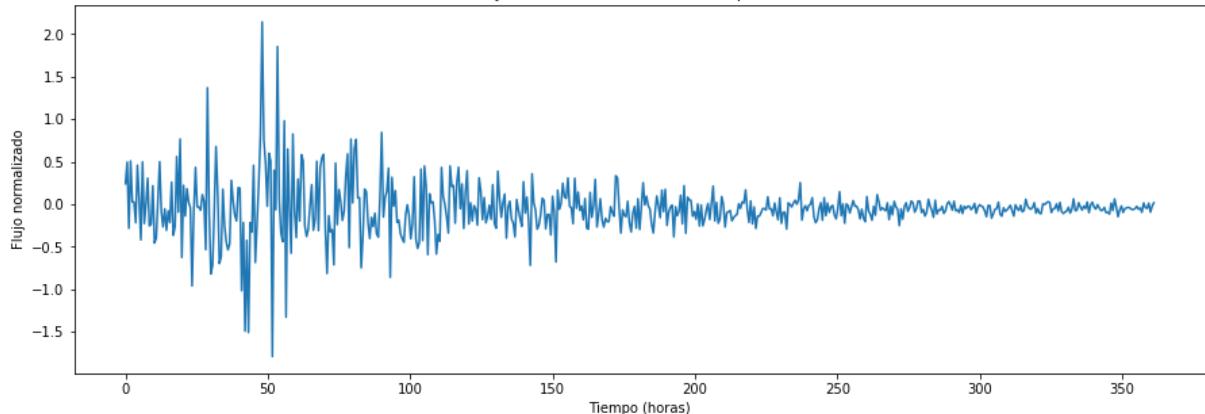
Flujo de la estrella nº 9 CON exoplanetas



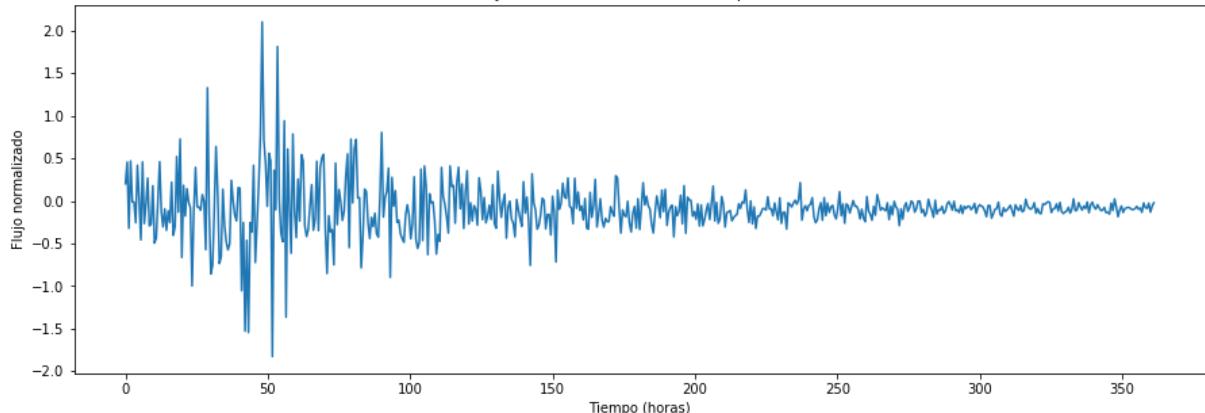
Flujo de la estrella nº 9 CON exoplanetas



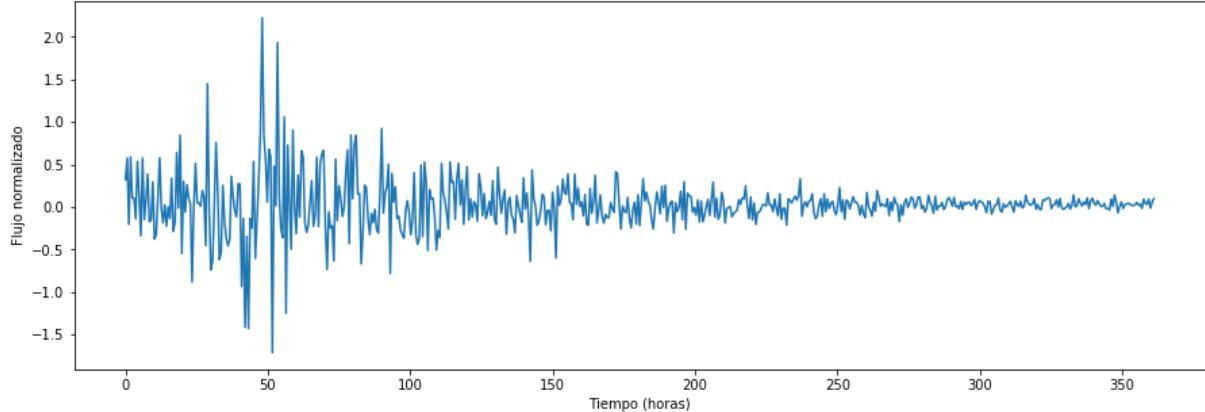
Flujo de la estrella nº 9 CON exoplanetas



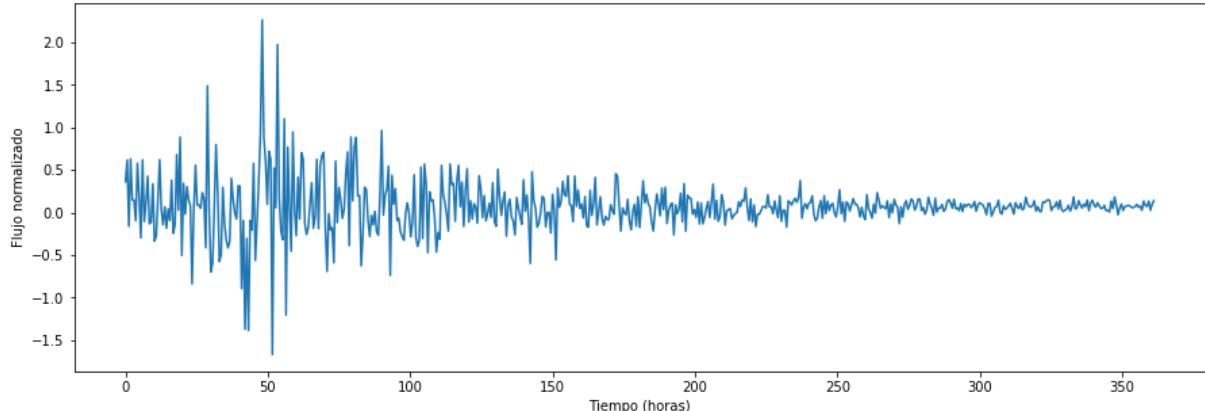
Flujo de la estrella nº 9 CON exoplanetas



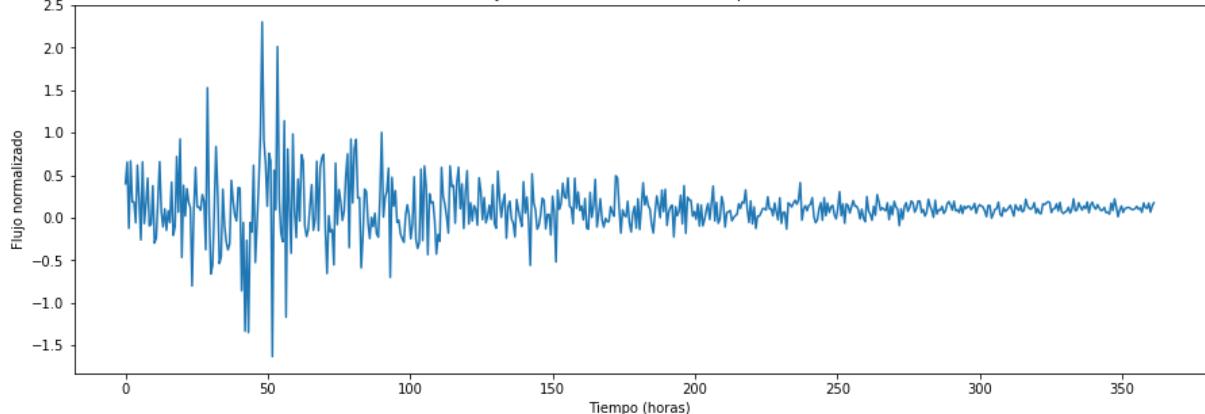
Flujo de la estrella nº 9 CON exoplanetas



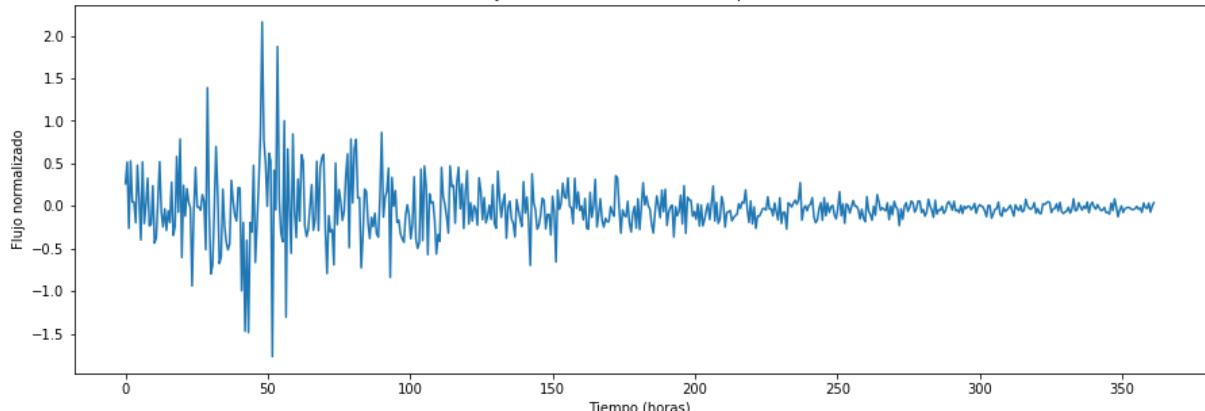
Flujo de la estrella nº 9 CON exoplanetas



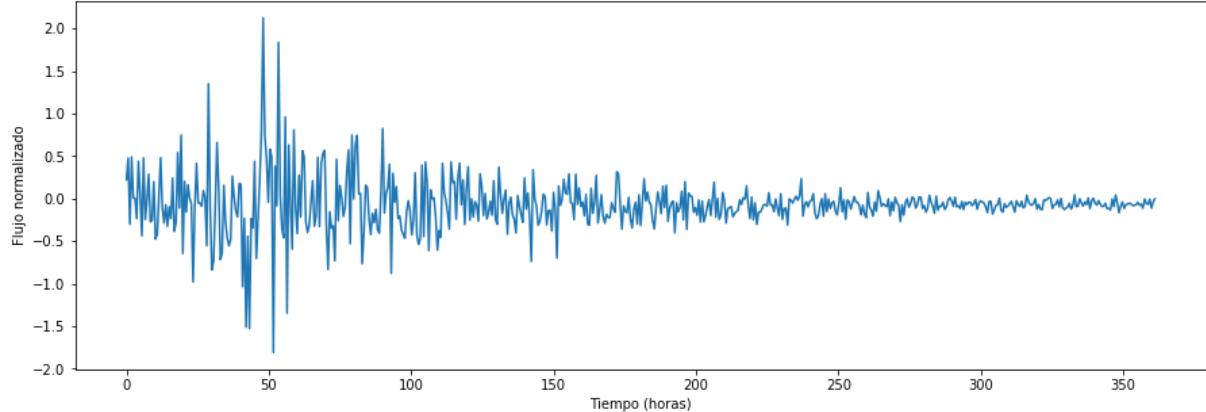
Flujo de la estrella nº 9 CON exoplanetas



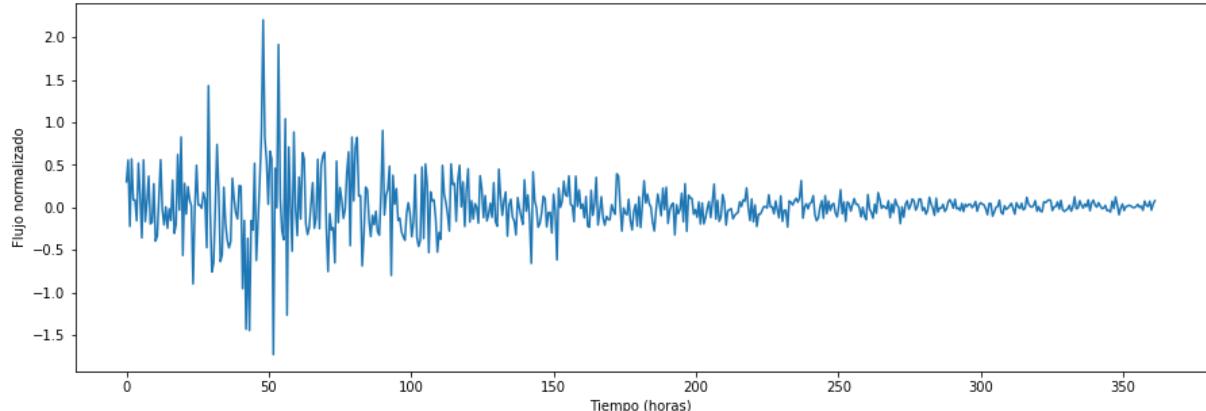
Flujo de la estrella nº 9 CON exoplanetas



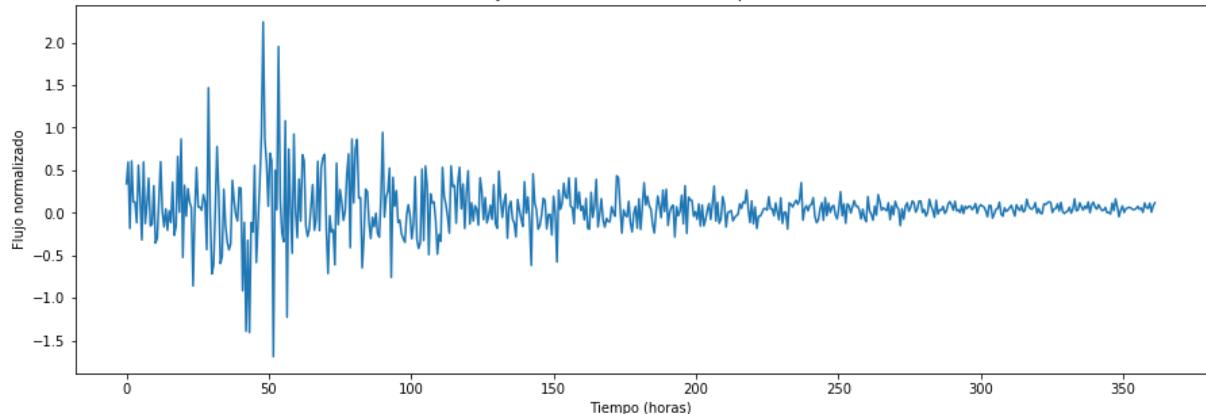
Flujo de la estrella nº 9 CON exoplanetas



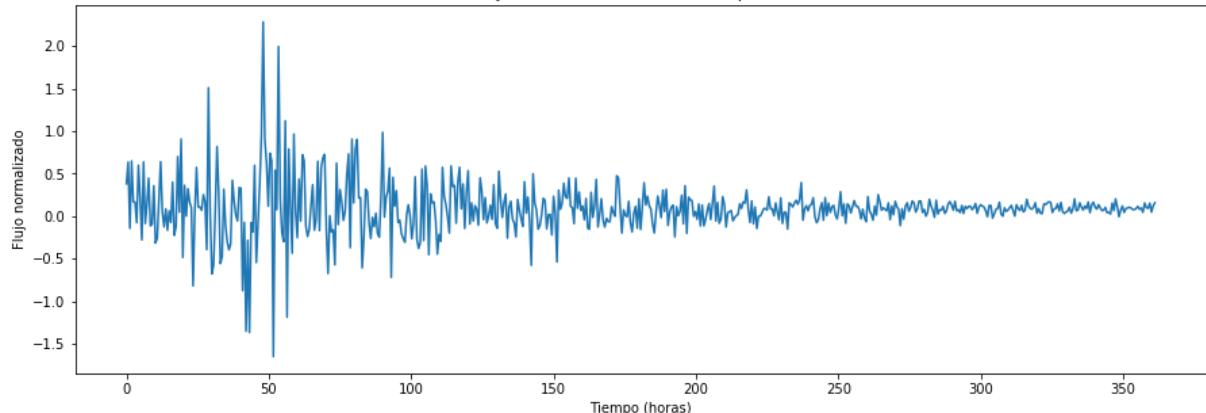
Flujo de la estrella nº 9 CON exoplanetas



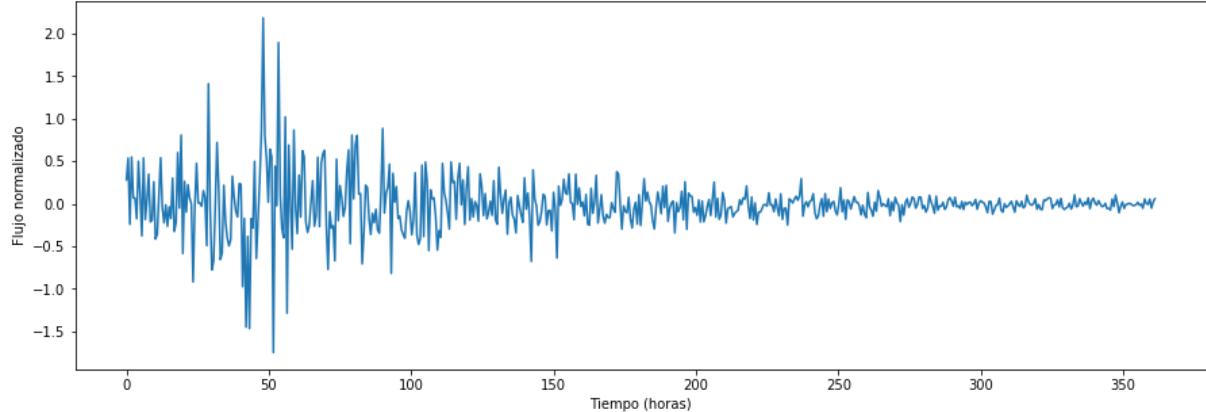
Flujo de la estrella nº 9 CON exoplanetas



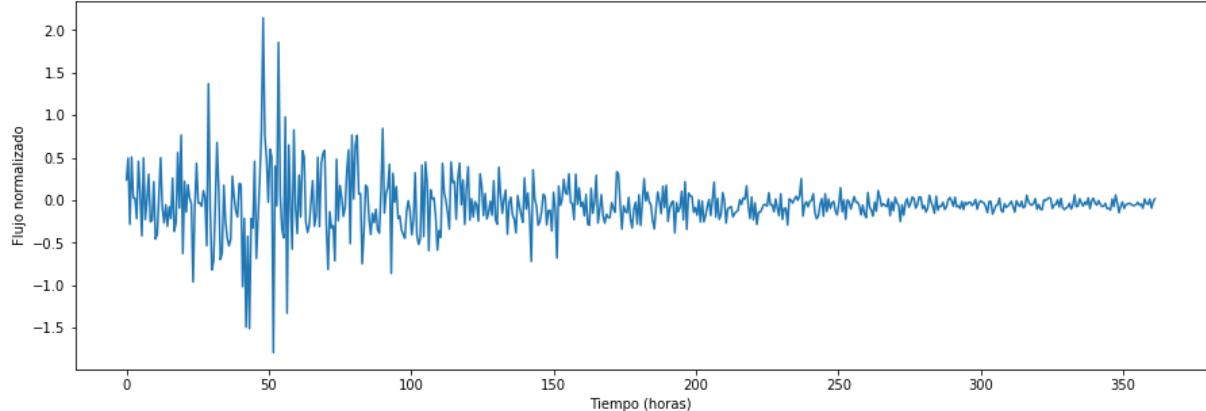
Flujo de la estrella nº 9 CON exoplanetas



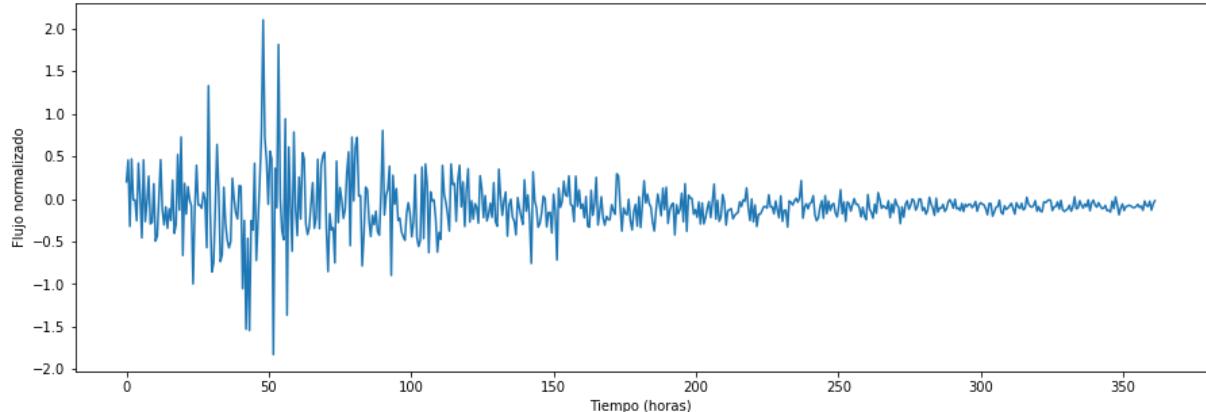
Flujo de la estrella nº 9 CON exoplanetas



Flujo de la estrella nº 9 CON exoplanetas



Flujo de la estrella nº 9 CON exoplanetas



Generar dataset de entrenamiento

Tras realizar el primer tratamiento de datos, PCA y las dos técnicas de *data augmentation*, el siguiente paso es generar el *dataset* de entrenamiento con todos los datos

```
In [26]: label = [2] * len(flujos_exo_da)
flujos_exo_da.insert(0, 'LABEL', label, True) #Insertar la columna 'LABEL' con todos los valores igual a '2'
flujos_exo_da
```

Out[26]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.5
0	2	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	...	0.1564
1	2	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	...	-0.0492
2	2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	...	-0.0359
3	2	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	...	0.0278
4	2	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	...	-0.0850
...
2844	2	0.023684	-0.143729	-0.088381	-0.066401	-0.111587	-0.160832	-0.117576	-0.114774	-0.082425	...	-0.1063
2845	2	0.032063	-0.235370	-0.018679	-0.038811	-0.059494	-0.147944	-0.141696	-0.180650	-0.026279	...	-0.0867
2846	2	-0.075598	0.080178	-0.234849	0.076402	-0.104667	-0.166866	-0.212455	-0.049279	-0.116961	...	-0.1329
2847	2	-0.108298	-0.179033	-0.191851	-0.137490	0.094292	-0.133475	-0.086013	-0.034150	-0.148294	...	-0.1271
2848	2	-0.123608	-0.048203	0.055908	0.105944	-0.156517	-0.122955	-0.360031	0.065988	-0.046692	...	-0.1282

2849 rows × 604 columns

```
In [27]: kepler_train_data_final = pd.DataFrame(flujos_exo_da) #Instanciar nuevo DataFrame con los datos de las estrellas con exoplanetas
kepler_train_data_final = kepler_train_data_final.append(kepler_train_data_pca[kepler_train_data_pca.LABEL == 1].iloc[:, :], ignore_index = True) #Añadir los datos de las estrellas sin exoplanetas
kepler_train_data_final
```

Out[27]:

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.5
0	2	0.064657	0.096775	-0.025112	-0.065888	-0.005493	-0.128811	0.052731	0.180928	0.435949	...	0.1564
1	2	0.280902	0.051640	0.132471	-0.162766	-0.081271	-0.007710	-0.063518	0.267813	0.160459	...	-0.0492
2	2	-0.095377	0.212582	-0.554484	0.243481	0.200693	0.329420	-0.143710	0.496918	-0.051265	...	-0.0359
3	2	-0.436844	0.222025	-0.404767	0.269585	-0.149680	0.317270	-0.205330	0.543809	0.133126	...	0.0278
4	2	0.156385	0.089766	-0.235699	-0.101338	-0.154256	0.326667	-0.209556	0.383505	-0.040992	...	-0.0850
...
7894	1	-3.936119	0.222828	-0.168616	-0.652407	-0.568182	2.094444	-0.296745	-1.226294	0.457405	...	-0.0183
7895	1	-0.091921	1.186475	1.311370	0.323385	-0.594974	-0.652316	0.471931	1.509093	1.447173	...	-0.0575
7896	1	0.114565	0.210029	0.394792	0.090145	0.045819	-0.217128	-0.025548	0.043999	-0.194226	...	-0.0175
7897	1	-0.004102	-0.161846	0.041366	0.126275	-0.002037	-0.070416	0.018862	0.026613	0.005338	...	0.0002
7898	1	0.320166	-0.382206	-0.145770	-0.418198	0.047609	-0.114099	0.096102	-0.066002	0.185191	...	-0.0393

7899 rows × 604 columns

Arriba se muestra el dataset final con el que se entrenará el modelo. Los datos han sido tratados; sus dimensiones, reducidas; y su desbalanceamiento, corregido. Contiene un total de 7899 estrellas, de las cuales 2849 tienen exoplanetas -'LABEL' = '2'- y 5050 no tienen exoplanetas -'LABEL' = '1'-.

Modelo de predicción

El modelo de predicción consistirá en una red neuronal artificial entrenada con los datos de entrenamiento finales -tras el primer tratamiento, reducción de dimensiones y *data augmentation*- y probada con los datos de test, que se están conformados por 565 no exoplanetas y 5 exoplanetas.

```
In [71]: #Cargar en variables distintas los datos de entrenamiento y test, separando las dimensiones de la clase
x_entreno = kepler_train_data_final.drop('LABEL', axis = 1)
y_entreno = kepler_train_data_final['LABEL']
x_test = kepler_test_data_pca.drop('LABEL', axis = 1)
y_test = kepler_test_data_pca['LABEL']
```

Construcción y entrenamiento de la red

```
In [80]: #Instanciar modelo secuencial: el output de cada capa creada es el input de la siguiente
modelo = Sequential()

#Capa neuronal oculta tipo dense -con todas las conexiones entre neuronas- con las dimensiones de entrada del dataset de entrenamiento y 100 dimensiones de salida
modelo.add(Dense(100, input_dim=x_entreno.shape[1]))

#Capa neuronal de salida con 100 dimensiones de entrada y 1 de salida -clase binaria-, con la función de activación 'sigmoid'
modelo.add(Dense(1, activation='sigmoid'))

#Compilar el modelo con la loss function "binary_crossentropy" ya que la clasificación es binaria
modelo.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

#Entrenar el modelo que estos parámetros y los datos de validación de test
modelo.fit(x_entreno, y_entreno, batch_size=20, epochs=10, validation_data=(x_test, y_test))
```

Train on 7899 samples, validate on 570 samples
Epoch 1/10
7899/7899 [=====] - 1s 92us/step - loss: -4.2711 - accuracy: 0.5930 - val_loss: 0.0597 - val_accuracy: 0.9807
Epoch 2/10
7899/7899 [=====] - 1s 66us/step - loss: -5.7133 - accuracy: 0.6341 - val_loss: 0.0088 - val_accuracy: 0.9825
Epoch 3/10
7899/7899 [=====] - 1s 67us/step - loss: -5.7404 - accuracy: 0.6383 - val_loss: -0.0089 - val_accuracy: 0.9825
Epoch 4/10
7899/7899 [=====] - 1s 69us/step - loss: -5.7454 - accuracy: 0.6391 - val_loss: -0.0143 - val_accuracy: 0.9825
Epoch 5/10
7899/7899 [=====] - 1s 70us/step - loss: -5.7456 - accuracy: 0.6391 - val_loss: -0.0175 - val_accuracy: 0.9825
Epoch 6/10
7899/7899 [=====] - 1s 69us/step - loss: -5.7457 - accuracy: 0.6391 - val_loss: -0.0203 - val_accuracy: 0.9842
Epoch 7/10
7899/7899 [=====] - 1s 71us/step - loss: -5.7459 - accuracy: 0.6391 - val_loss: -0.0151 - val_accuracy: 0.9825
Epoch 8/10
7899/7899 [=====] - 1s 69us/step - loss: -5.7478 - accuracy: 0.6391 - val_loss: -0.0207 - val_accuracy: 0.9842
Epoch 9/10
7899/7899 [=====] - 1s 71us/step - loss: -5.7479 - accuracy: 0.6392 - val_loss: -0.0231 - val_accuracy: 0.9842
Epoch 10/10
7899/7899 [=====] - 1s 80us/step - loss: -5.7479 - accuracy: 0.6392 - val_loss: -0.0252 - val_accuracy: 0.9860

```
Out[80]: <keras.callbacks.callbacks.History at 0x2cc94dab748>
```

Predicción y precisión

```
In [81]: predicción = modelo.predict_classes(x_test) #Crear predicción
predicción = np.where(predicción == 0, 2, predicción) #Ajustar valores para que concuerden con los
del dataset

#Crear la matriz de confusión
cm = confusion_matrix(y_test, predicción)
cm = {'Real: SIN exoplanetas': cm[0], 'Real: CON exoplanetas': cm[1]}
cm = pd.DataFrame.from_dict(cm, orient = 'index', columns = ['Predicción: SIN exoplanetas', 'Predicción: CON exoplanetas'])

#Medir la precisión del modelo
print("La precisión total del modelo es del: {}%".format(round(accuracy_score(y_test,predicción),
4)*100))
cm
```

La precisión total del modelo es del: 98.95%

Out[81]:

	Predicción: SIN exoplanetas	Predicción: CON exoplanetas
Real: SIN exoplanetas	562	3
Real: CON exoplanetas	3	2

El modelo ha logrado predecir 2 de los 5 exoplanetas, y 562 de los 565 no exoplanetas. La precisión total de las predicciones del modelo es del 98.95%