Hunting Exoplanets In Space - Scatter & Line Plots

```
1 # Load the training dataset.
2 import pandas as pd
3 exo_train_df=pd.read_csv("/content/exoTrain.csv")
4 exo_train_df.head()
5
```

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLU
0	2	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-7
1	2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-8
2	2	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	43
3	2	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	31
4	2	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-98

5 rows × 3198 columns

```
1 # Display Statistical information of the train dataset
2 exo_train_df.describe()
3
4
```

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	
count	728.000000	728.000000	728.000000	728.000000	728.000000	
mean	1.050824	-66.847239	-106.061346	-99.817335	-118.512500	
std	0.219790	9911.491996	9585.760746	8892.230967	8762.188487	
min	1.000000	-216192.000000	-219338.000000	-212822.000000	-217468.000000	-22
25%	1.000000	-67.035000	-70.557500	-70.220000	-64.437500	
50%	1.000000	0.285000	-1.150000	0.690000	-2.290000	
75%	1.000000	68.205000	54.835000	53.882500	48.910000	
max	2.000000	150725.800000	129578.360000	102184.980000	82253.980000	6

8 rows × 3197 columns

```
1 # Check the number of rows and columns in the 'exo_train_df'.
2 exo_train_df.shape
3
```

```
(4292, 3198)
```

Check For The Missing Values

```
1 # Find the total number of missing values in the 'exo_train_df'.
2 exo_train_df.isnull().sum()
3
```

```
LABEL        0
FLUX.1       0
FLUX.2       0
FLUX.3       0
FLUX.4       0
            ..
FLUX.3193    1
FLUX.3194    1
FLUX.3195    1
FLUX.3196    1
FLUX.3197    1
Length: 3198, dtype: int64
```

There are no missing values in the DataFrame.

Slicing A DataFrame Using The `iloc[]` Function

Create Pandas series for the first 3 stars and the last 3 stars in the DataFrame.

Syntax:

`dataframe_name.iloc[row_position_start : row_position_end, column_position_start : column_position_end]`

In this syntax:

- `row_position_start` denotes the position of the row in the DataFrame **starting** from whose values you want to take in the new Pandas series or DataFrame.
- `row_position_end` denotes the position of the row in the DataFrame till whose values you want to take in the new Pandas series or DataFrame.
- `column_position_start` denotes the position of the column in the DataFrame **starting** from whose values you want to take in the new Pandas series or DataFrame.
- `column_position_end` denotes the position of the column in the DataFrame till whose values you want to take in the new Pandas series or DataFrame.

You can verify manually whether we have extracted the values from the first row or not by

```
# Create a Pandas series for the first star and store it in a variable called 'star_0'.
star_0 = exo_train_df.iloc[0, :]
star_0.head()
```

```
LABEL         2
FLUX.1    93.85
FLUX.2    83.81
FLUX.3     20.1
FLUX.4   -26.98
Name: 0, dtype: object
```

```
type(star_0)
```

```
pandas.core.series.Series
```

```
# Create a Pandas series for the second star and store it in a variable called 'star_1'
star_1 =exo_train_df.iloc[1, :]
star_1.head()
```

```
LABEL         2
FLUX.1   -38.88
FLUX.2   -33.83
FLUX.3   -58.54
FLUX.4   -40.09
Name: 1, dtype: object
```

```
# Create a Pandas series for the third star and store it in a variable called 'star_2'.
star_2 =exo_train_df.iloc[2, :]
star_2.head()
```

```
LABEL          2
FLUX.1    532.64
FLUX.2    535.92
FLUX.3    513.73
FLUX.4    496.92
Name: 2, dtype: object
```

```
# Create a Pandas series for the last star and store it in a variable called 'star_5086
star_5086 =exo_train_df.iloc[-1, :]
star_5086.head()
```

```
LABEL     1.00
FLUX.1    0.77
FLUX.2   -4.73
FLUX.3    3.75
```

```
FLUX.4   -0.87
Name: 4291, dtype: float64
```

Scatter and Line Plots of Flux

Now plot the **Flux** values on the $y-axis$ for each observation for a star. On $x-axis$, we will plot numbers ranging from `1` to `3197`.

Scatter And Line Plots For First 3 Stars^

To make this plot,

1. We first need to import a Python module named `matplotlib.pyplot` with `plt` as an alias. This module is exclusively designed for creating graphs such as bar graphs, histogram, line plot, scatter plot etc. We will learn more about this module as we go on in this course.

```
import matplotlib.pyplot as plt
```

2. Then we need to call the `figure()` function from the `plt` module to resize the plot. The `figure()` function takes `figsize=(horizontal_width, vertical_height)` parameter as an input.

```
plt.figure(figsize=(16, 4))
```

3. Then we need either a Python list, a NumPy array or a Pandas series containing the numbers between `1` and `3197` to plot them on the $x-axis$.

```
x_values_star_0 = np.arange(1, 3198)
```

4. Then we need `star_0` Pandas series to plot the `FLUX` values on the $y-axis$ for the first star in the DataFrame.

```
y_values_star_0 = star_0[1:]
```

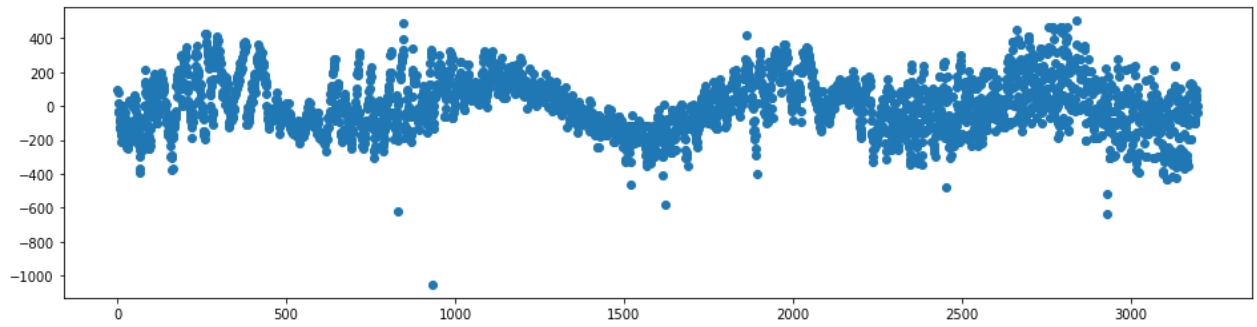
5. Then we need to call the `scatter()` function from the `plt` module with the required inputs as described in the third and the fourth steps.

```
plt.scatter(x_values_star_0, y_values_star_0)
```

6. Finally, we need to call the `show()` function from the `plt` module.

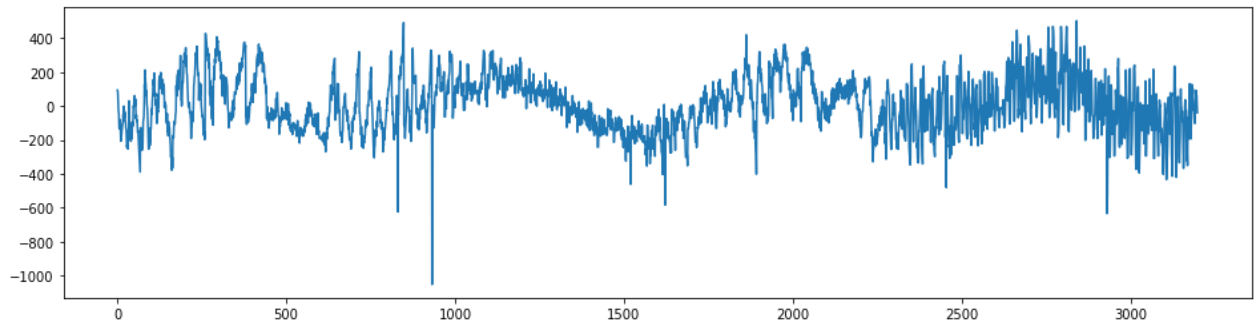
```
plt.show()
```

```
#Create a scatter plot for 'star_0' Pandas series.
# 1. Import the 'numpy' and 'matplotlib.pyplot' modules.
import numpy as np
import matplotlib.pyplot as plt


# 2. Call the 'figure()' function to resize the plot.
plt.figure(figsize=(16,4))


# Here, 16 means the plot is 16 units wide and 4 units high. Play with these numbers to
# Call the 'scatter()' function to make a scatter plot between the x and y values.
# The scatter() function requires two inputs: x and y where x is the data to be plotted
# In our case, x is a Pandas series of numbers between 1 and 3197 and y is the 'FLUX' v


# Here, star_0[1:] is a Pandas series containing all the 'FLUX' values starting from th
# The 'arange(1, 3198)' function from the 'numpy' module will generate numbers from 1 t
# 3. Call the 'scatter()' function.
x_values_star_0 = np.arange(1,3198)
y_values_star_0= star_0[1:]
plt.scatter(x_values_star_0, y_values_star_0)


# 4. Call the 'show()' function.
plt.show()

# The 'show()' function displays the plot.
```



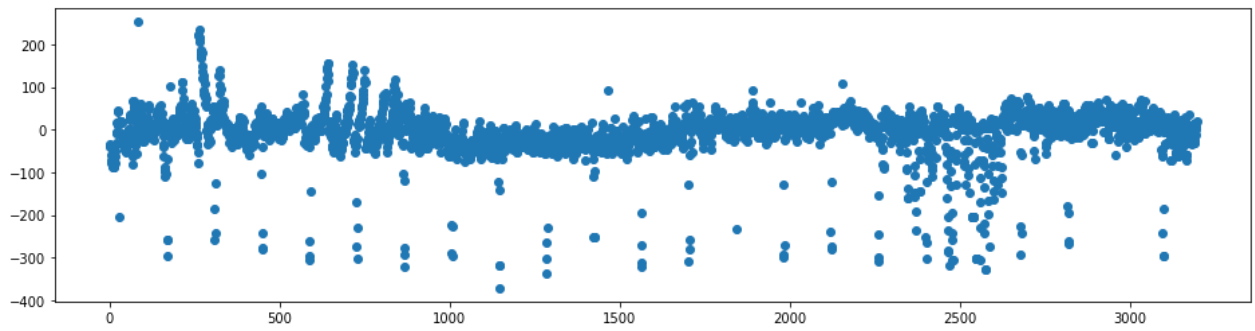

```
# Create a line plot for 'star_0' Pandas series.
# Line plot for the first star in the DataFrame.


# Call the plot(x, y) function to draw a line plot between the x and y values.
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(16,4))
```

```
 9 x_values_star_0 = np.arange(1,3198)
10 y_values_star_0= star_0[1:]
11
12 plt.plot(x_values_star_0, y_values_star_0)
13 plt.show()
14
15
16
```



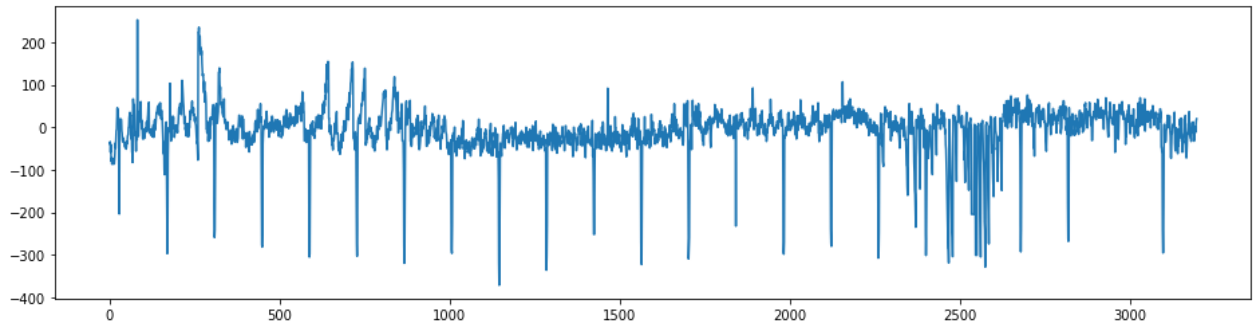
The line plot also confirms the periodic downward-peaks in the `FLUX` values.

```
 1 # Create a scatter plot for the second star, i.e., 'star_1'.
 2
 3 plt.figure(figsize=(16,4))
 4 star_1 =exo_train_df.iloc[1, :]
 5 star_1.head()
 6
 7 x_values_star_1 = np.arange(1,3198)
 8 y_values_star_1= star_1[1:]
 9
10 plt.scatter(x_values_star_1, y_values_star_1)
11 plt.show()
12
13
14
15
16
17
```



It is quite difficult to spot any clear pattern in the scatter plot for the second star in the DataFrame. Let's draw a line plot to identify a pattern.

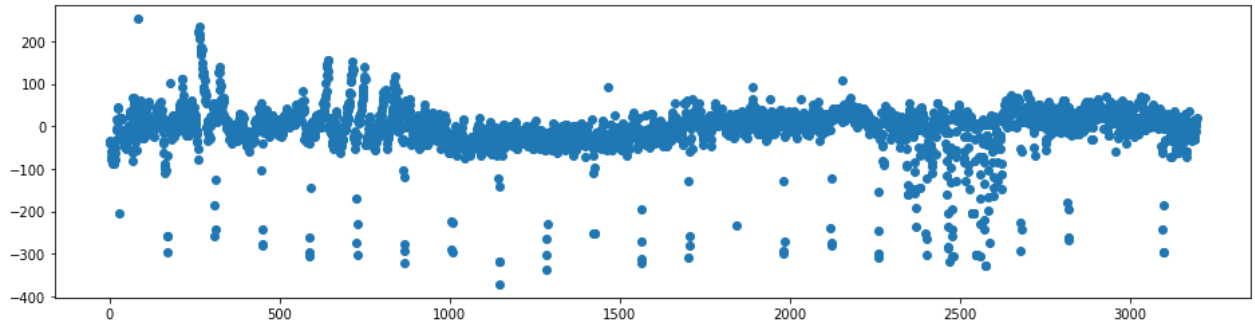
```
# Create a line plot for the second star, i.e., 'star_1'.
plt.figure(figsize=(16,4))
star_1 =exo_train_df.iloc[1, :]
star_1.head()

x_values_star_1 = np.arange(1,3198)
y_values_star_1= star_1[1:]


plt.plot(x_values_star_1, y_values_star_1)
plt.show()
```



As we can see, there are consistent sudden drops in the brightness levels for the second star in the DataFrame. This suggests that the planet is orbiting its star at very high radial speed. Also, the planet could be very close to the star.

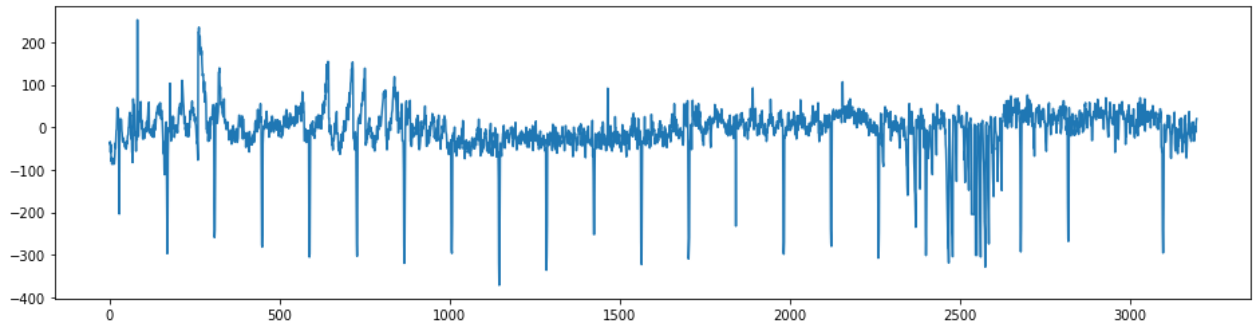
```
#Create a scatter plot for the third star, i.e., 'star_2'.

plt.figure(figsize=(16,4))
star_2 =exo_train_df.iloc[1, :]
star_2.head()

x_values_star_2 = np.arange(1,3198)
y_values_star_2= star_2[1:]

plt.scatter(x_values_star_2, y_values_star_2)
plt.show()
```

```
plt.show()
```



Here also, we can spot a clear repetitive downward-peaks which confirms that the star has at least one planet.

```
#Create a line plot for the third star, i.e, 'star_2'.
plt.figure(figsize=(16,4))
star_2 =exo_train_df.iloc[1, :]
star_2.head()

x_values_star_2 = np.arange(1,3198)
y_values_star_2= star_2[1:]


plt.plot(x_values_star_2, y_values_star_2)
plt.show()
```

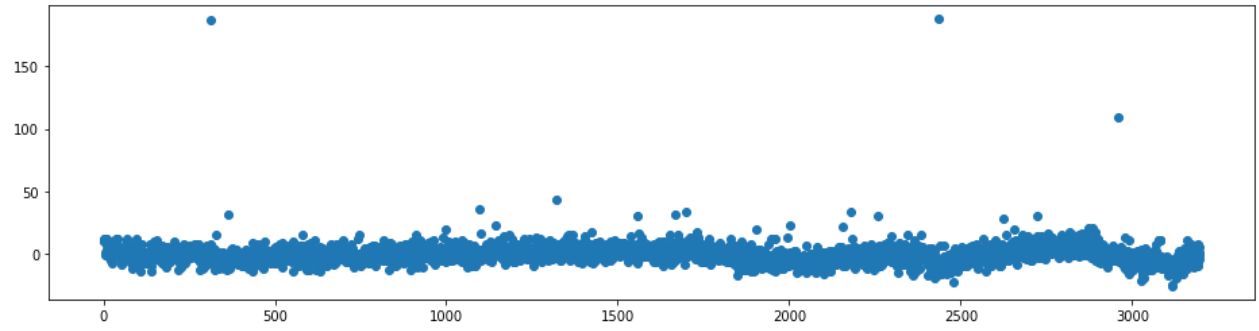


The line plot also confirms the repetitive downward-peak pattern.

Scatter Plots And Line Plots For 2nd Last Star^^^

Now, create the scatter plots and line plots for 2nd Last Star or any star in the DataFrame which have been labelled or classified as `1`.

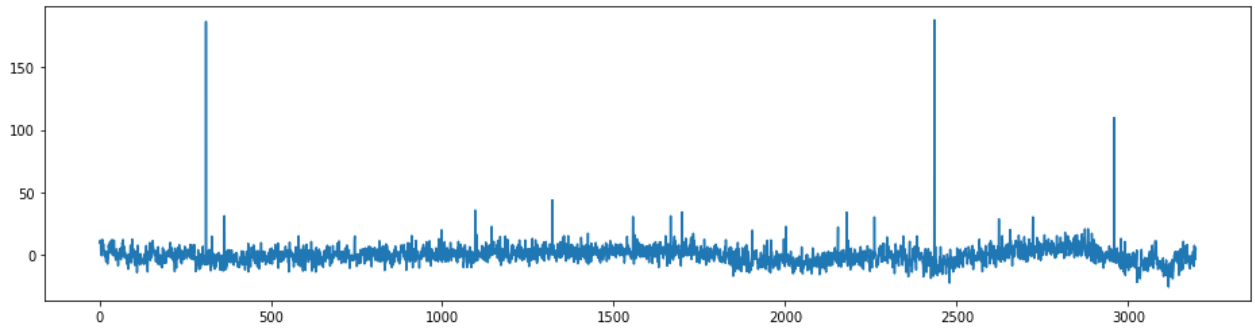
```
# Create a scatter plot for the second-last star, i.e., 'star_5085' in the DataFrame.
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(16,4))
star_5085 =exo_train_df.iloc[-2, :]
star_5085.head()
x_values_star_5085 = np.arange(1,3198)
y_values_star_5085= star_5085[1:]

plt.scatter(x_values_star_5085, y_values_star_5085)
plt.show()


```



There is no clear periodic downward-peak pattern in the `FLUX` values for the second-last star.

```
# Student Action: Create a line plot for the second-last star in the DataFrame.
plt.figure(figsize=(16,4))
star_5085 =exo_train_df.iloc[-2, :]
star_5085.head()
x_values_star_5085 = np.arange(1,3198)
y_values_star_5085= star_5085[1:]

plt.plot(x_values_star_5085, y_values_star_5085)
plt.show()

```

11



The line-plot also confirms that there is no clear periodic downward-peak pattern in the `FLUX` values.