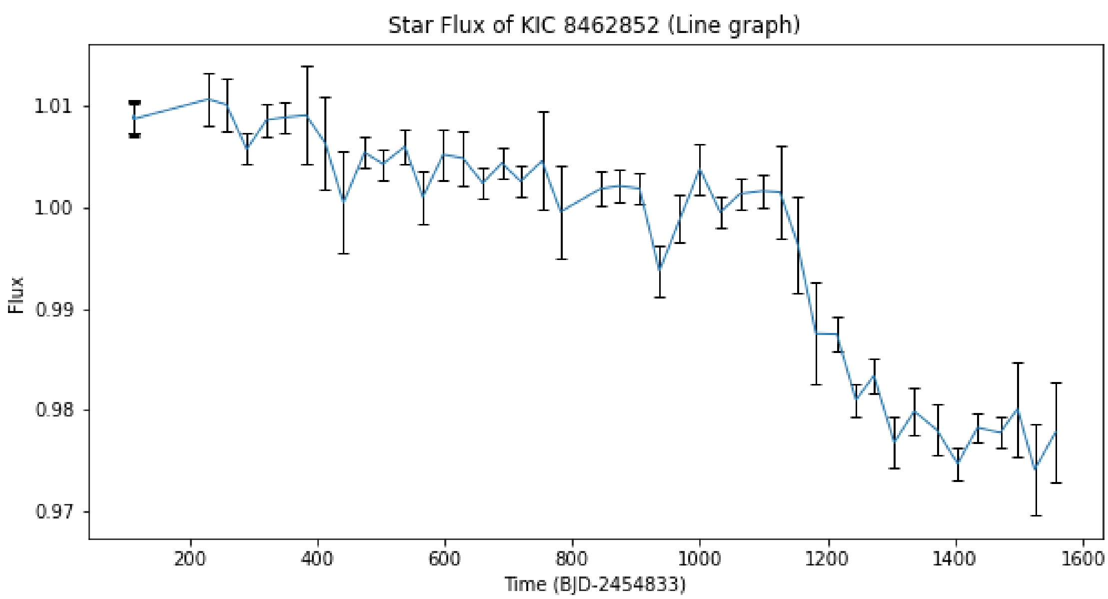
Star Flux of KIC 8462852

April 22, 2023

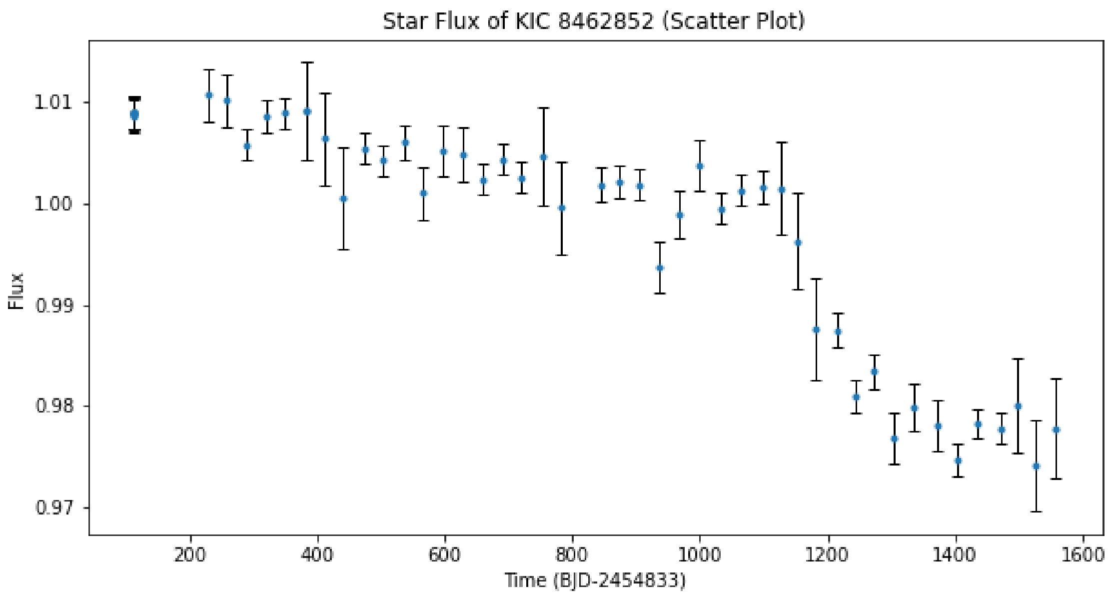
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
```

[2]:
```
data = np.loadtxt("PHYS267_DataSet_Astrophysics_StarFlux_Full.csv",␣
 ↪delimiter=",", skiprows=1)
time = data[:,0]
flux = data[:,1]
uncertainty = data[:,2]
orientation = data[:,3]
```

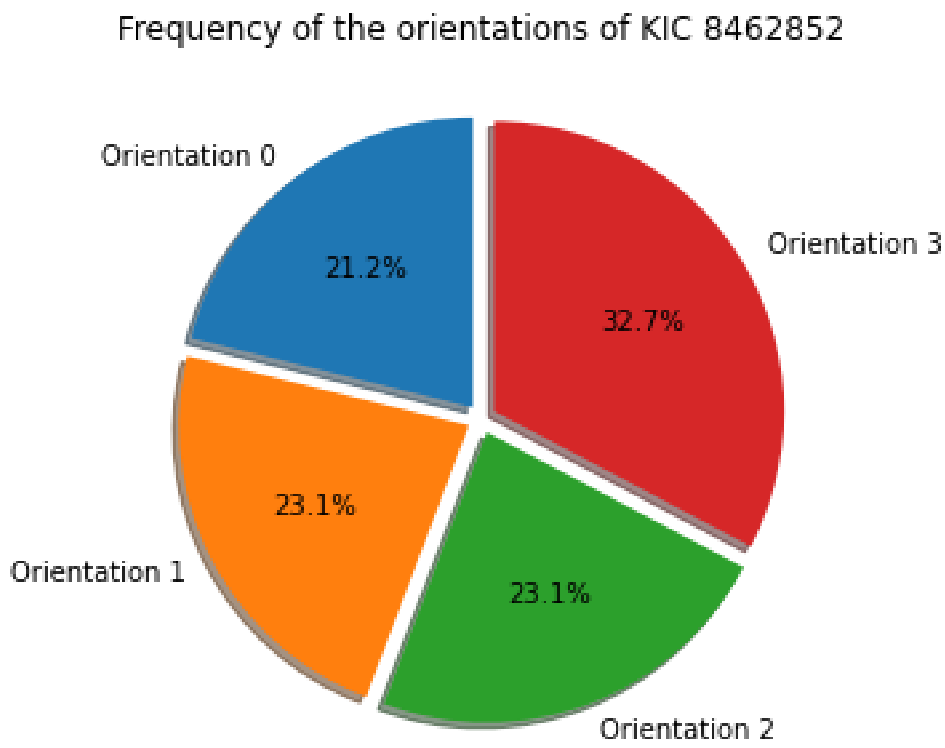
[3]:
```
# Line graph of all flux + uncertainty
plt.figure(figsize=(10,5))
plt.errorbar(x=time, y=flux, yerr=uncertainty, ecolor='black', linewidth=1,␣
 ↪capsize=3)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 (Line graph)")
plt.show()
```



[4]:
```
# Scatter plot of all flux + uncertainty
plt.figure(figsize=(10,5))
plt.errorbar(x=time, y=flux, fmt=".", yerr=uncertainty, ecolor='black',␣
↪linewidth=1, capsize=3)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 (Scatter Plot)")
plt.show()
```

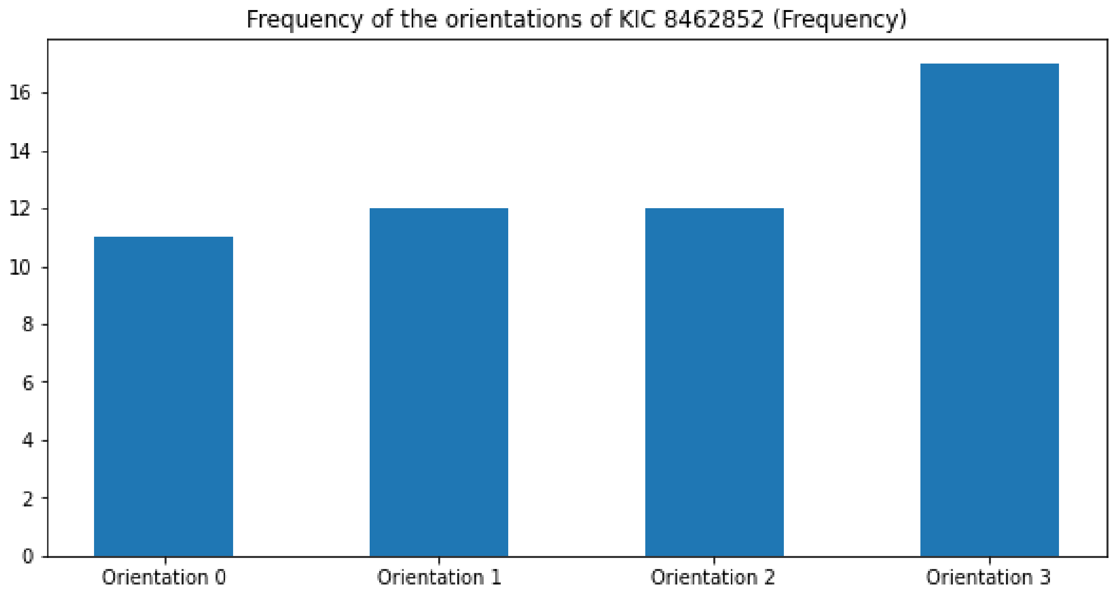


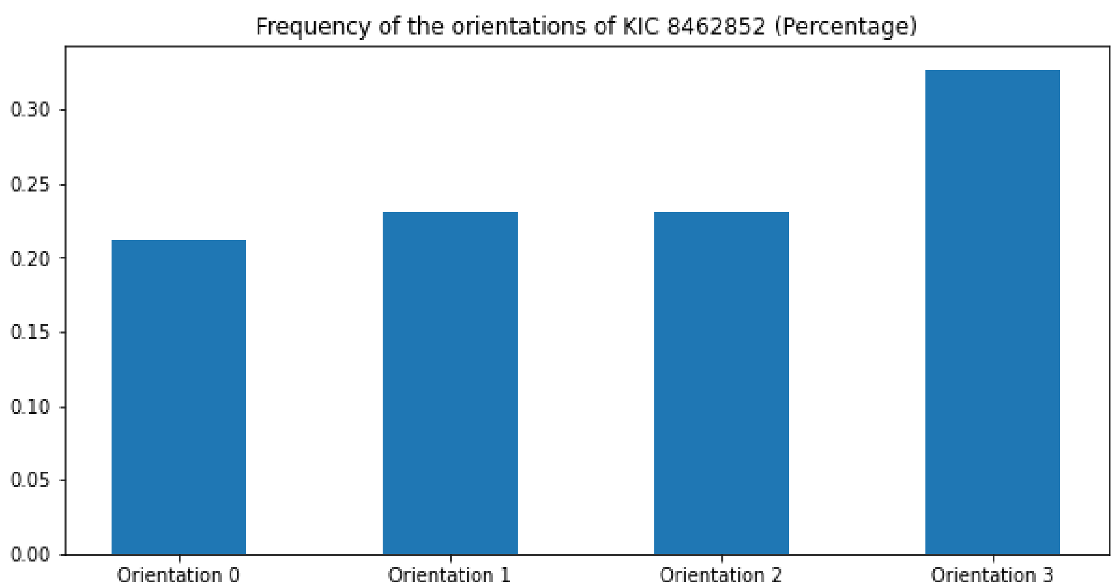
[5]:
```
# Pie chart for the orientation category
labels = "Orientation 0", "Orientation 1", "Orientation 2", "Orientation 3"
sizes = [0, 0, 0, 0]
explode = (0.05, 0.05, 0.05, 0.05)
for i in orientation:
    sizes[int(i)] += 1

plt.figure(figsize=(10,5))
plt.pie(sizes, labels=labels, explode=explode, autopct='%1.1f%%', shadow=True,␣
↪startangle=90)
plt.title("Frequency of the orientations of KIC 8462852")
plt.show()
```



[6]:

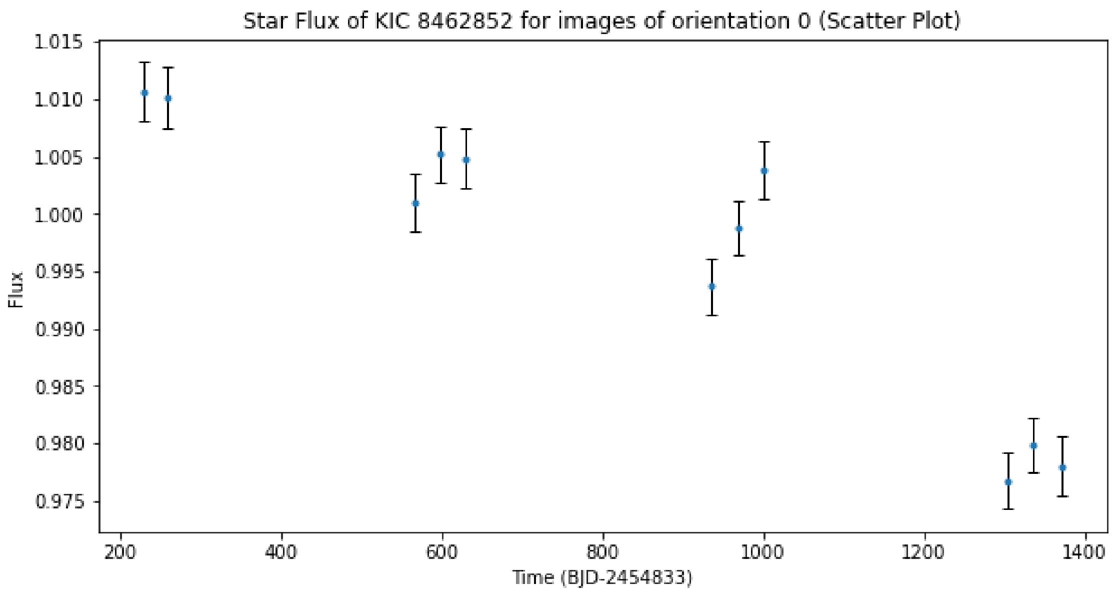
```
# Bar chart for the orientation category
labels = "Orientation 0", "Orientation 1", "Orientation 2", "Orientation 3"
sizes = [0, 0, 0, 0]
for i in orientation:
    sizes[int(i)] += 1

plt.figure(figsize=(10,5))
plt.bar(x=labels, height=sizes, width=0.5)
plt.title("Frequency of the orientations of KIC 8462852 (Frequency)")
plt.show()

plt.figure(figsize=(10,5))
plt.bar(x=labels, height=np.array(sizes)/52, width=0.5)
plt.title("Frequency of the orientations of KIC 8462852 (Percentage)")
plt.show()
```



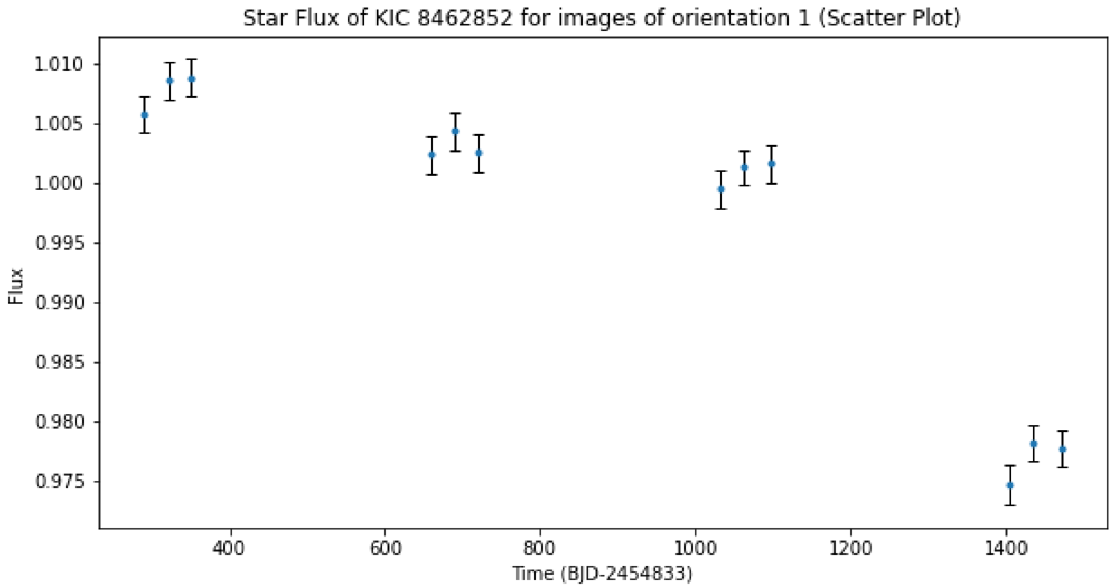


[7]:
```
# Scatter plot and uncertainty for all orientation 0 images
time_orientation0 = []
flux_orientation0 = []
uncertainty_orientation0 = []
for i in range(len(orientation)):
    if int(orientation[i]) == 0:
        time_orientation0.append(time[i])
        flux_orientation0.append(flux[i])
```

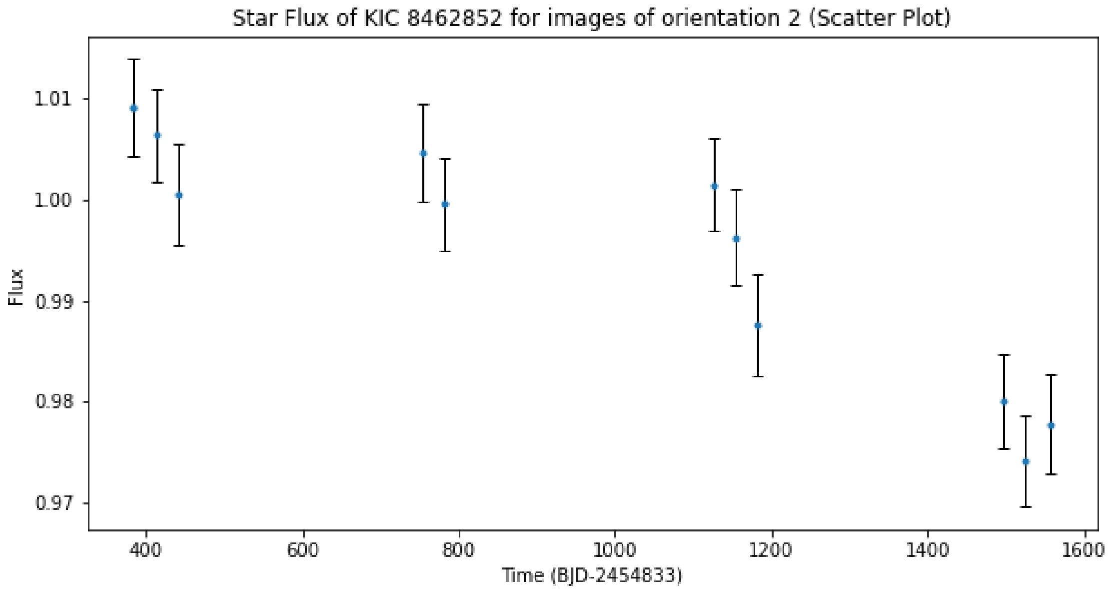
```
        uncertainty_orientation0.append(uncertainty[i])

plt.figure(figsize=(10,5))
plt.errorbar(x=time_orientation0, y=flux_orientation0, fmt=".",␣
 ↪yerr=uncertainty_orientation0, ecolor='black', linewidth=1, capsize=3)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 for images of orientation 0 (Scatter Plot)")
plt.show()
```



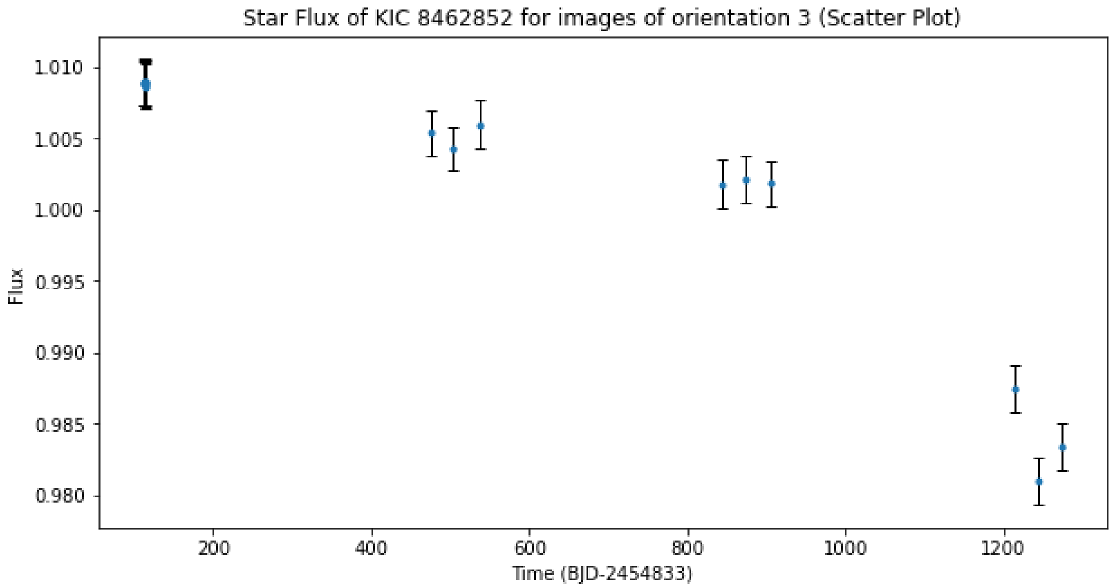
[8]:
```
# Scatter plot and uncertainty for all orientation 1 images
time_orientation1 = []
flux_orientation1 = []
uncertainty_orientation1 = []
for i in range(len(orientation)):
    if int(orientation[i]) == 1:
        time_orientation1.append(time[i])
        flux_orientation1.append(flux[i])
        uncertainty_orientation1.append(uncertainty[i])

plt.figure(figsize=(10,5))
plt.errorbar(x=time_orientation1, y=flux_orientation1, fmt=".",␣
 ↪yerr=uncertainty_orientation1, ecolor='black', linewidth=1, capsize=3)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 for images of orientation 1 (Scatter Plot)")
plt.show()
```



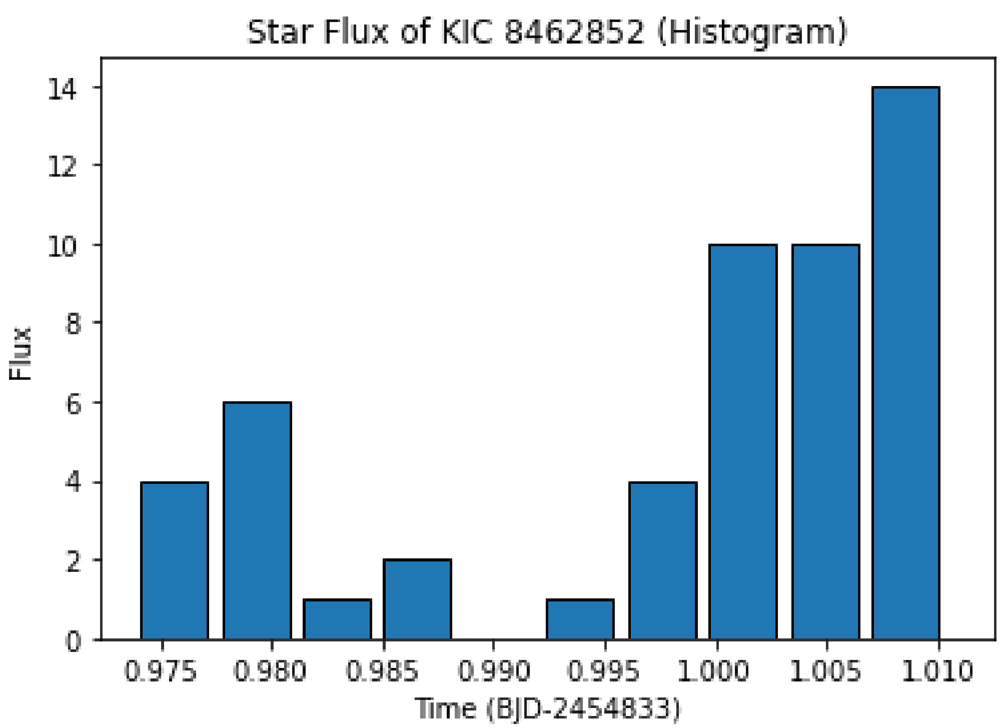
[9]:
```python
# Scatter plot and uncertainty for all orientation 2 images
time_orientation2 = []
flux_orientation2 = []
uncertainty_orientation2 = []
for i in range(len(orientation)):
    if int(orientation[i]) == 2:
        time_orientation2.append(time[i])
        flux_orientation2.append(flux[i])
        uncertainty_orientation2.append(uncertainty[i])

plt.figure(figsize=(10,5))
plt.errorbar(x=time_orientation2, y=flux_orientation2, fmt=".",␣
↪yerr=uncertainty_orientation2, ecolor='black', linewidth=1, capsize=3)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 for images of orientation 2 (Scatter Plot)")
plt.show()
```



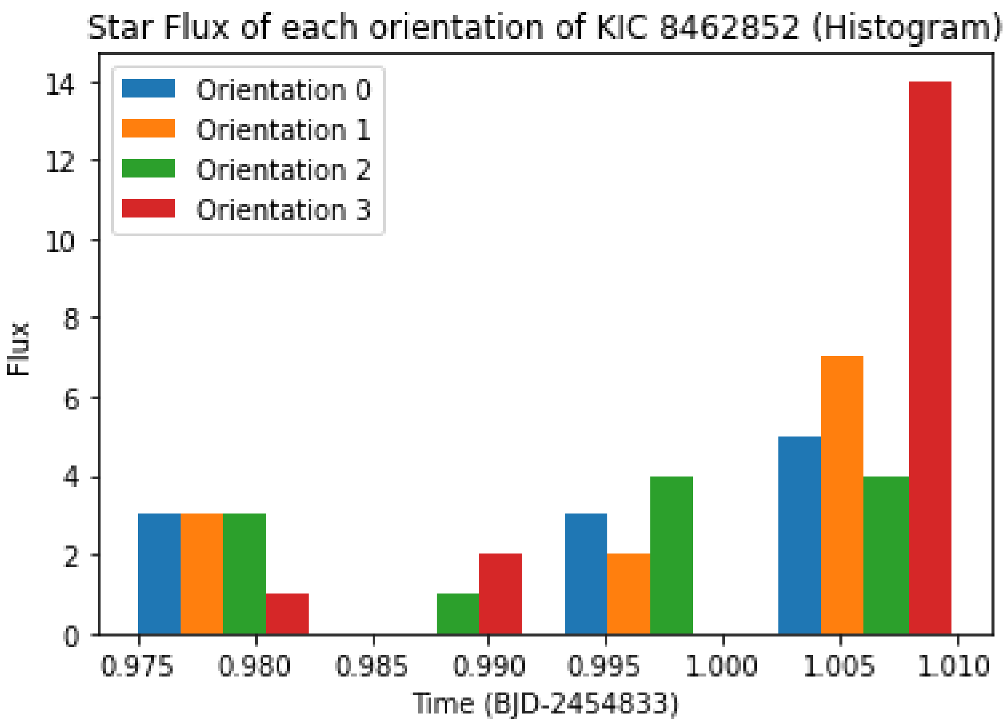
[10]:
```python
# Scatter plot and uncertainty for all orientation 3 images
time_orientation3 = []
flux_orientation3 = []
uncertainty_orientation3 = []
for i in range(len(orientation)):
    if int(orientation[i]) == 3:
        time_orientation3.append(time[i])
        flux_orientation3.append(flux[i])
        uncertainty_orientation3.append(uncertainty[i])

plt.figure(figsize=(10,5))
plt.errorbar(x=time_orientation3, y=flux_orientation3, fmt=".",␣
 ↪yerr=uncertainty_orientation3, ecolor='black', linewidth=1, capsize=3)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 for images of orientation 3 (Scatter Plot)")
plt.show()
```



[11]:
```python
# Histogram for all flux
plt.hist(x=flux, bins=10, edgecolor='black', width=0.003)
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of KIC 8462852 (Histogram)")
plt.show()
```

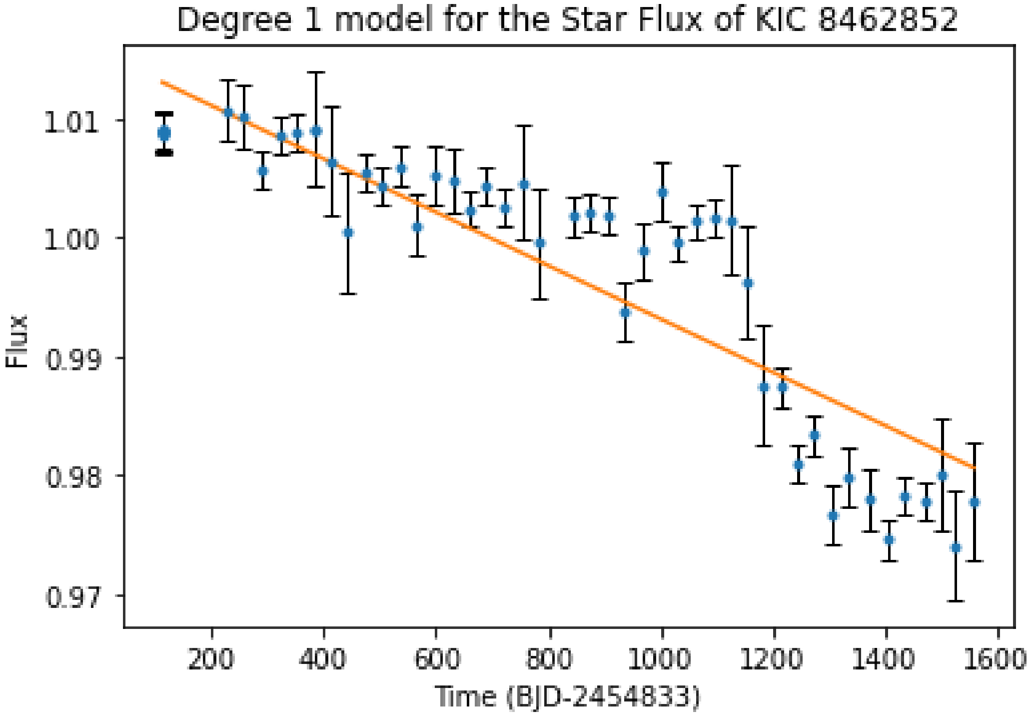


[12]:
```
# Histogram for all flux in each orientation
plt.hist([flux_orientation0, flux_orientation1, flux_orientation2,␣
 ↪flux_orientation3], bins=4,
         label=['Orientation 0', 'Orientation 1', 'Orientation 2', 'Orientation␣
 ↪3'])
plt.xlabel("Time (BJD-2454833)")
plt.ylabel("Flux")
plt.title("Star Flux of each orientation of KIC 8462852 (Histogram)")
plt.legend(loc='upper left')
plt.show()
```

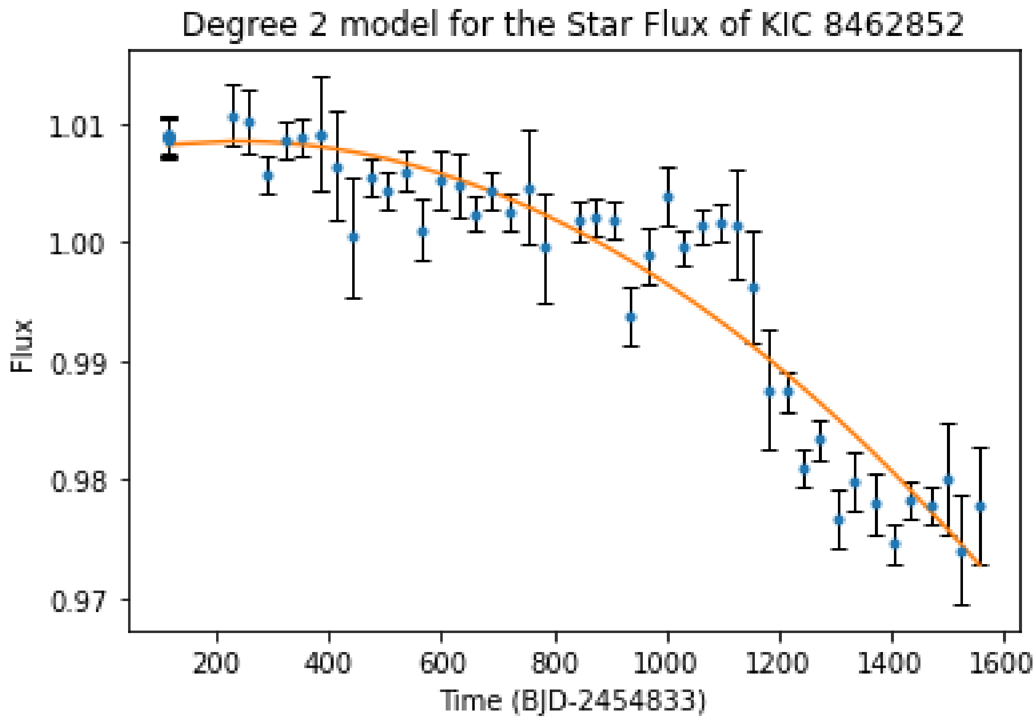


[13]:
```
# Find a model for the datapoints
model_limit = 0.95
r2 = 0
degree = 1
x = np.array(time)
y = np.array(flux)

while r2 < model_limit:
    # Create model
    parameters = np.polyfit(x, y, degree)
```

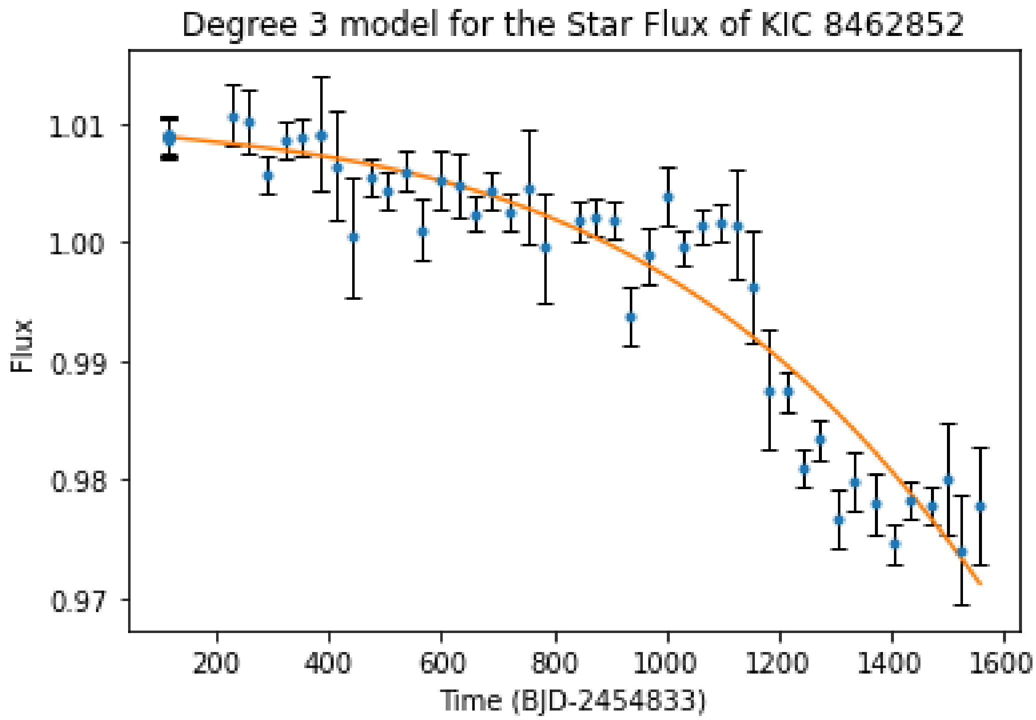
```
    powers = list(reversed(range(degree+1)))
    model = []
    for xi in x:
        yi = 0
        for i in range(len(powers)):
            yi += parameters[i] * xi**(powers[i])
        model.append(yi)
    # Plot model
    plt.errorbar(x=x, y=y, fmt=".", yerr=uncertainty, ecolor='black',␣
↪linewidth=1, capsize=3)
    plt.plot(x, model)
    plt.xlabel("Time (BJD-2454833)")
    plt.ylabel("Flux")
    plt.title("Degree {} model for the Star Flux of KIC 8462852".format(degree))
    plt.show()
    # Print best fit parameters and R^2
    print("For degree {}, the best fit paramters are: {}".format(degree,␣
↪parameters))
    print("For degree {}, R^2 is: {}".format(degree, r2_score(y, model)))
  ␣
↪print("==========================================================================
    # Increase the degree and put R^2 value so the while loop can check if R^2␣
↪less than our desired R^2
    # and use a model with current degree+1
    r2 = r2_score(y, model)
    degree += 1
```



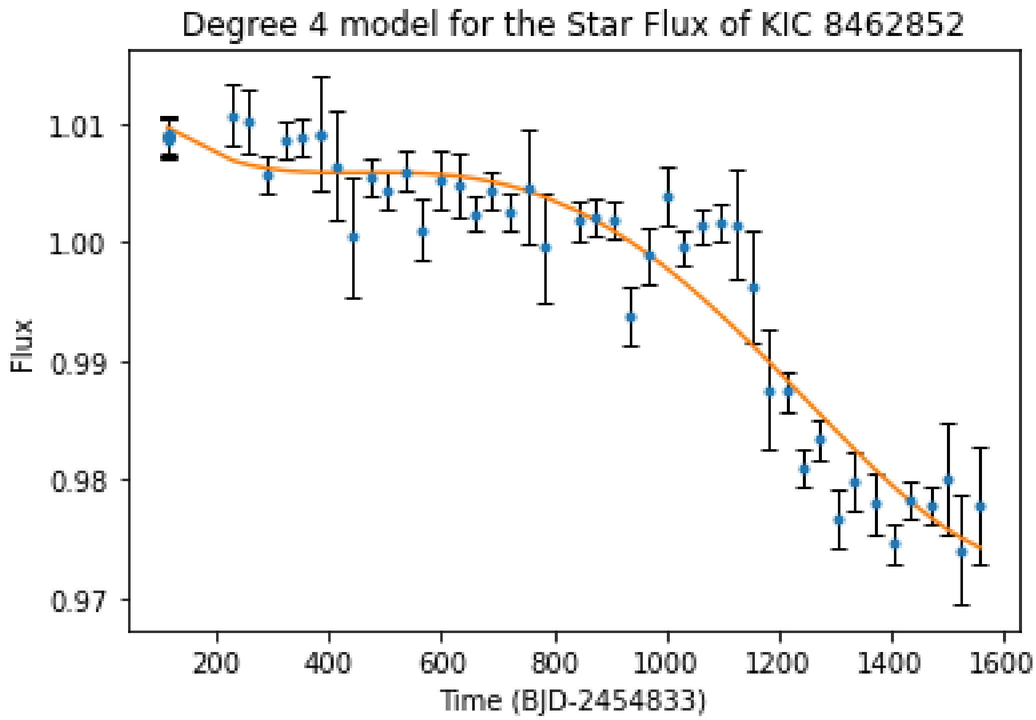

```
For degree 1, the best fit paramters are: [-2.24213105e-05  1.01556343e+00]
For degree 1, R^2 is: 0.7993336922893973
================================================================================
=======
```



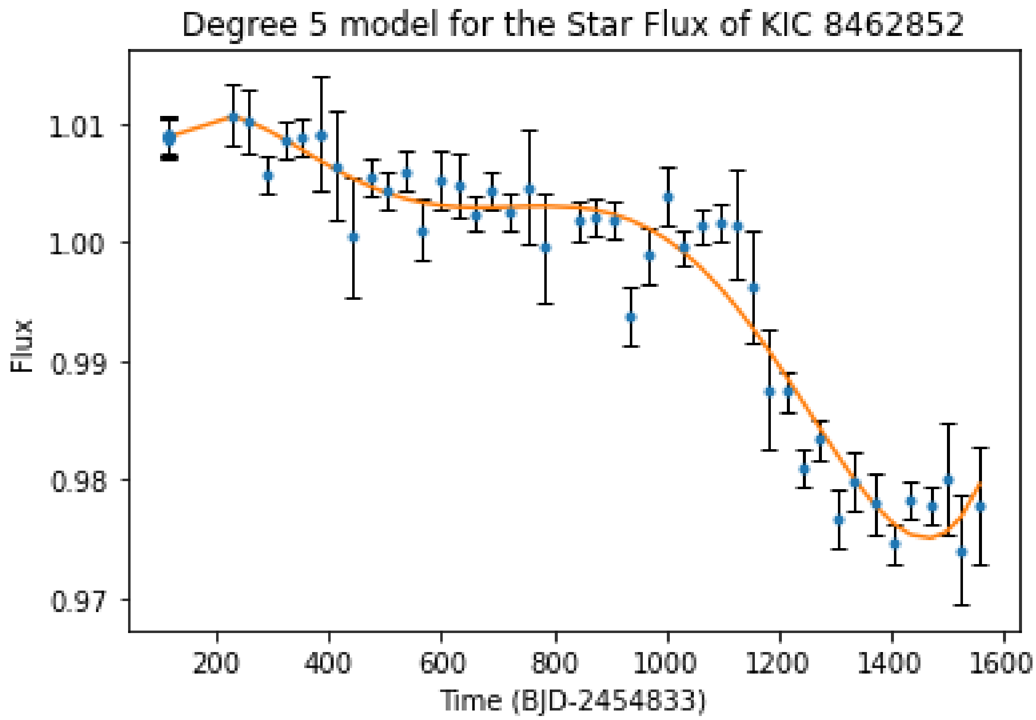

```
For degree 2, the best fit paramters are: [-2.03062552e-08  9.42627998e-06
1.00739054e+00]
For degree 2, R^2 is: 0.8980902144397775
===============================================================================
=======
```



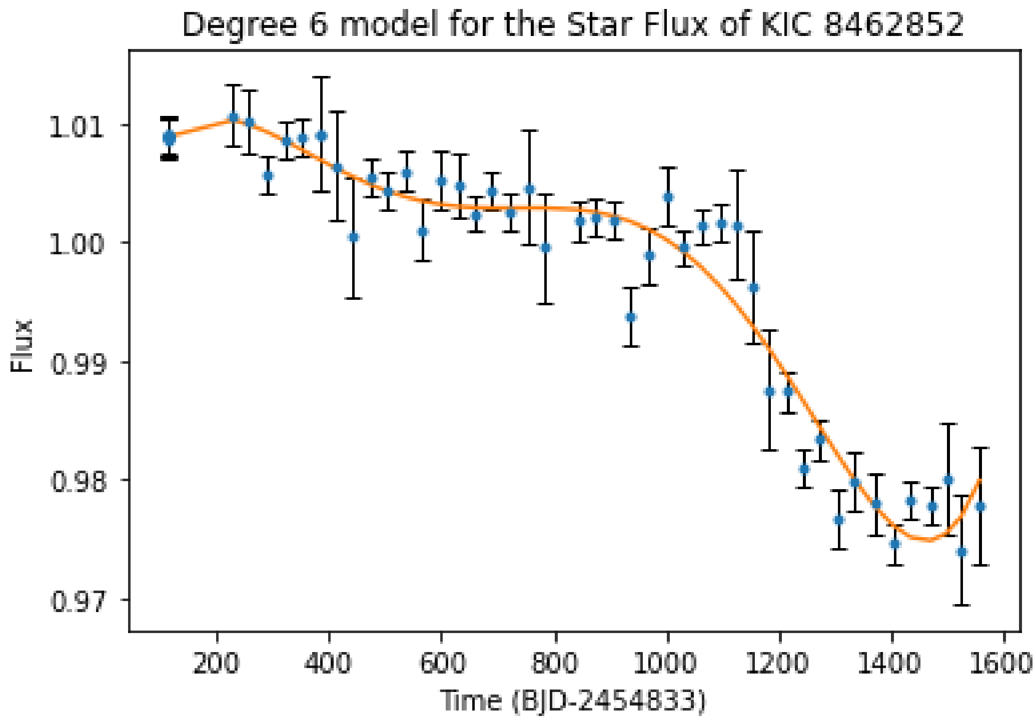

```
For degree 3, the best fit paramters are: [-9.29813444e-12  1.89098690e-09
-4.85763303e-06  1.00938594e+00]
For degree 3, R^2 is: 0.9011333779362698
================================================================================
=======
```



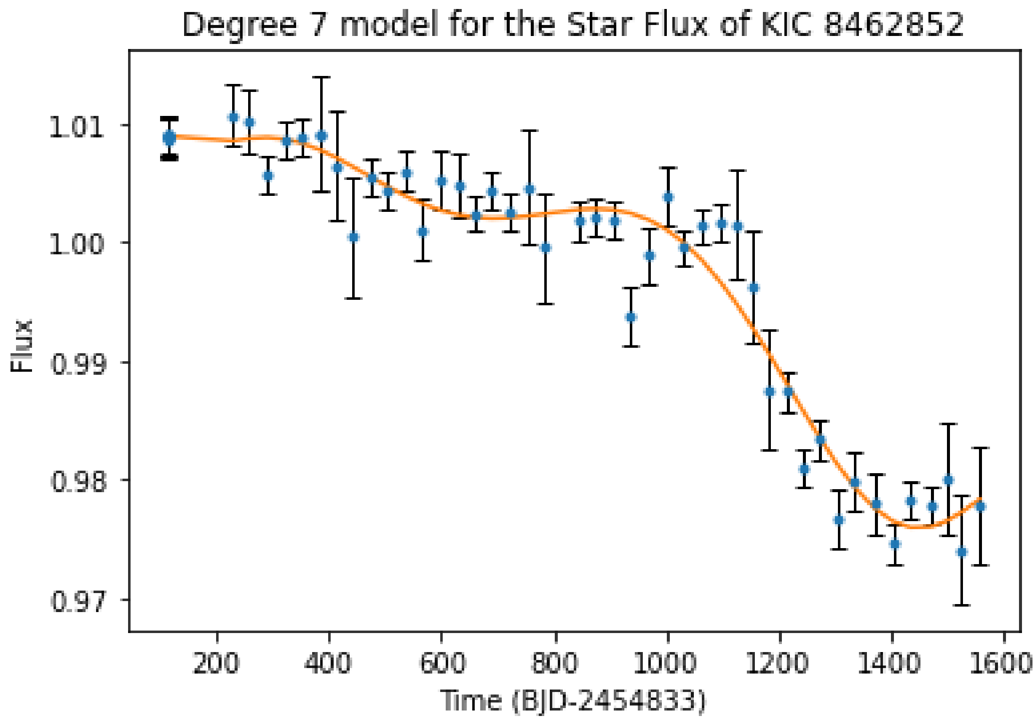

```
For degree 4, the best fit paramters are: [ 4.92232772e-14 -1.68316432e-10
1.69366694e-07 -6.78410380e-05
  1.01538209e+00]
For degree 4, R^2 is: 0.911669035268412
==============================================================================
=======
```



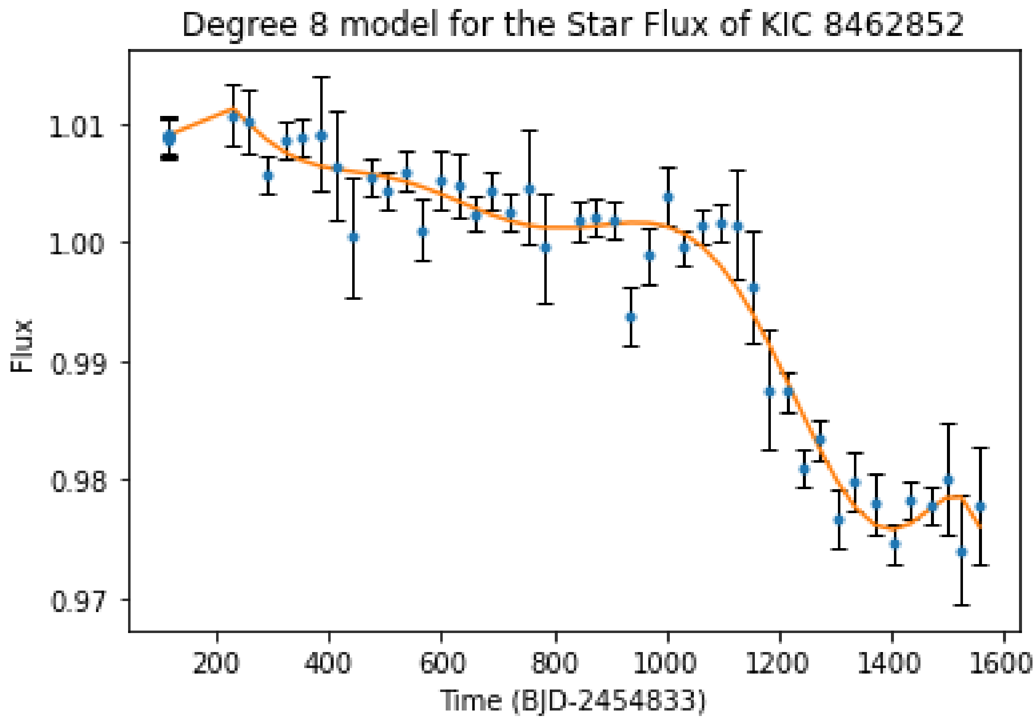

```
For degree 5, the best fit paramters are: [ 2.42515134e-16 -9.44641932e-13
1.30658334e-09 -7.85306729e-07
  1.84904009e-04  9.96180437e-01]
For degree 5, R^2 is: 0.9426870159344701
===============================================================================
=======
```




```
For degree 6, the best fit paramters are: [ 4.77119632e-20  4.71304190e-18
-4.87026804e-13  8.80378978e-10
 -5.88869357e-07  1.44476802e-04  9.98822402e-01]
For degree 6, R^2 is: 0.94282667352051
================================================================================
=======
```




```
For degree 7, the best fit paramters are: [-6.87543164e-22  4.08330576e-18
-9.54399095e-15  1.11660316e-11
 -6.90223491e-09  2.18536760e-06 -3.26087001e-04  1.02620098e+00]
For degree 7, R^2 is: 0.9463885025664199
================================================================================
=======
```




```
For degree 8, the best fit paramters are: [-3.27486544e-24  2.14005337e-20
-5.78183807e-17  8.37595548e-14
 -7.06208513e-11  3.51601748e-08 -9.96421983e-06  1.42502829e-03
  9.34347295e-01]
For degree 8, R^2 is: 0.955507457020038
================================================================================
=======
```

```
[ ]:
```