pulsar5

November 4, 2022

1 Pulsar Emission Data Analysis

2 All Imports that may or may not be needed and used for the notebook

[ ]:
```
#currently including any and all Imports that maybe needed for the project.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.feature_selection import RFE
import datetime as dt
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances
from scipy.cluster.hierarchy import linkage, dendrogram, cut_tree
from scipy.spatial.distance import pdist
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.dates as mdates
from scipy.stats import pearsonr
from scipy import stats
import statistics
import math
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.tsa.tsatools import lagmat
from numpy import array
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Bidirectional
```

3 Section for extracting from a tar file.

Currently implemented for original TAR File structure.

[ ]:
```
#This is also found in the main file under tarunzip.py
import tarfile
import os
import sys

#tar = tarfile.open("pulseTarFile.tar")
#tar.extractall('./Data')
#tar.close()
```

3.1 Beginning of Exploration

3.1.1 Examining the data

In this section we are determining the total integrity of the data to determine if further comprehensive data cleaning and uniforming processes are needed.

[ ]:
```
colnames = ['Pulse Number', 'Brightness', 'Uncertainty']
pulsar = pd.read_csv("Data/J1456-6843.pulses", sep = ' ', header = None, names␣
↪= colnames)
```

[ ]:
```
pulsar.shape
```

[ ]:
```
(1219, 3)
```

[ ]:
```
pulsar.head(25)
```

[ ]:

	Pulse Number	Brightness	Uncertainty
0	1	0.053904	0.005560
1	2	0.058653	0.004821
2	3	0.110208	0.005196
3	4	0.034716	0.004729
4	5	0.056101	0.004619
5	6	0.046168	0.005074
6	7	0.055648	0.004916
7	8	0.060890	0.004581
8	9	0.024388	0.004922
9	10	0.039370	0.004633
10	11	0.009141	0.004581
11	12	0.145273	0.005053
12	13	0.039953	0.004938
13	14	-0.002554	0.004409
14	15	0.035696	0.004903
15	16	0.046869	0.004706
16	17	0.082637	0.004596
17	18	0.349419	0.006828
18	19	0.058343	0.004650

```
19            20    0.090261     0.005068
20            21    0.120429     0.005141
21            22    0.209730     0.005389
22            23    0.088045     0.004945
23            24    0.203736     0.008553
24            25    0.024098     0.004641
```

[ ]:
```
pulsar.describe()
```

[ ]:
```
       Pulse Number   Brightness  Uncertainty
count   1219.000000  1219.000000  1219.000000
mean     610.000000     0.104176     0.005410
std      352.039297     0.081916     0.001282
min        1.000000    -0.007285     0.001075
25%      305.500000     0.045763     0.004728
50%      610.000000     0.081228     0.004966
75%      914.500000     0.144228     0.005541
max     1219.000000     0.825366     0.016201
```

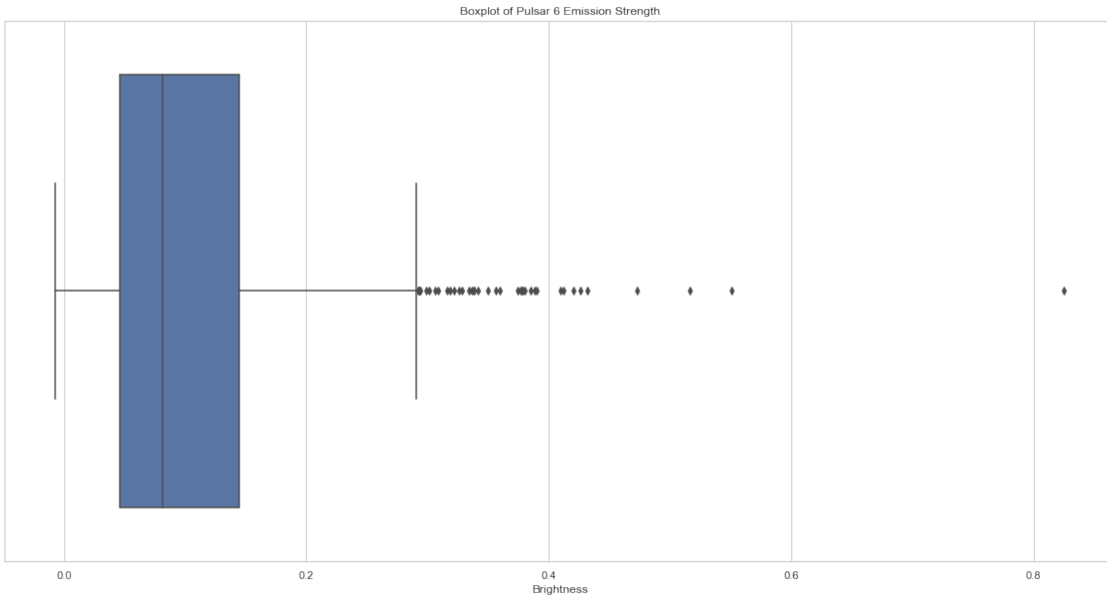
[ ]:
```
nullBoolBrightness = pd.isnull(pulsar["Brightness"])

pulsar[nullBoolBrightness]
```

[ ]:
```
Empty DataFrame
Columns: [Pulse Number, Brightness, Uncertainty]
Index: []
```

[ ]:
```
pulsar["Brightness"].describe()
```

[ ]:
```
count    1219.000000
mean        0.104176
std         0.081916
min        -0.007285
25%         0.045763
50%         0.081228
75%         0.144228
max         0.825366
Name: Brightness, dtype: float64
```

[ ]:
```
plt.figure(figsize=(20,10))
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=pulsar["Brightness"]).set_title("Boxplot of Pulsar 6␣
↪Emission Strength")
```



[ ]:
```
medianpulse6 = pulsar["Brightness"].median()
print("Median of Pulsar6: ", medianpulse6)
pulsar['Binary'] = np.where(pulsar['Brightness'] > medianpulse6, 1, 0)
```

```
Median of Pulsar6:  0.081228
```

[ ]:
```
pulsar
```

[ ]:

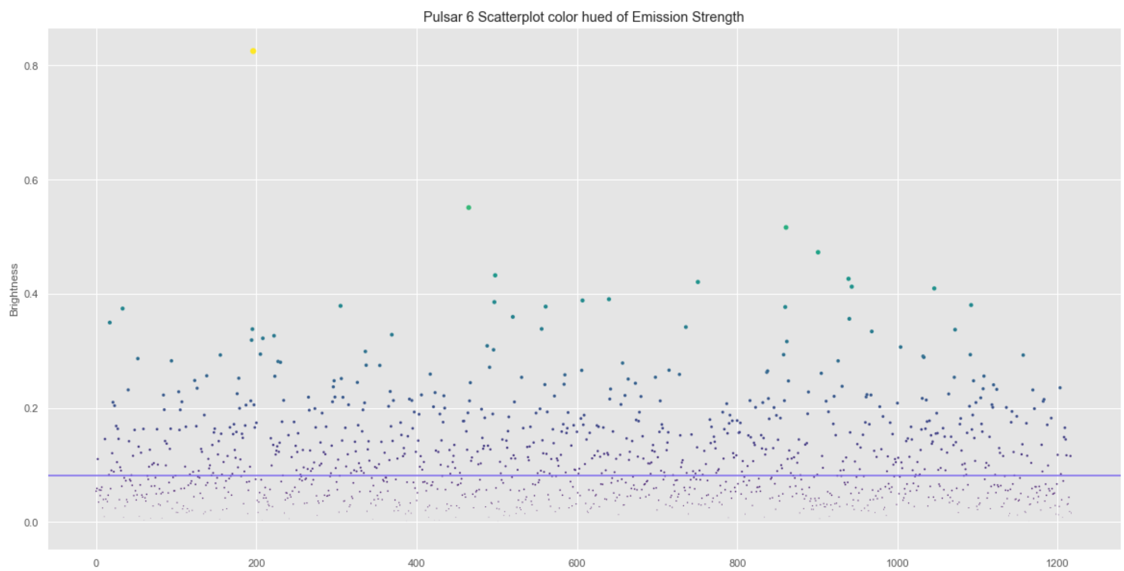
	Pulse Number	Brightness	Uncertainty	Binary
0	1	5.390386e-02	0.005560	0
1	2	5.865279e-02	0.004821	0
2	3	1.102083e-01	0.005196	1
3	4	3.471609e-02	0.004729	0
4	5	5.610133e-02	0.004619	0
…	…	…	…	…
1214	1215	4.321559e-02	0.004991	0
1215	1216	1.830750e-02	0.004578	0
1216	1217	1.155671e-01	0.005212	1
1217	1218	1.562609e-02	0.004686	0
1218	1219	-1.137418e-08	0.001075	0

[1219 rows x 4 columns]

[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = pulsar.Brightness.values
plt.style.use('ggplot')
```

```
ax = sns.scatterplot(data=pulsar["Brightness"], s= strength*50, c=strength,␣
↪cmap="viridis", marker="o").set_title('Pulsar 6 Scatterplot color hued of␣
↪Emission Strength')
ax= plt.axhline( y=0.081228, ls='-',c='mediumslateblue')
```

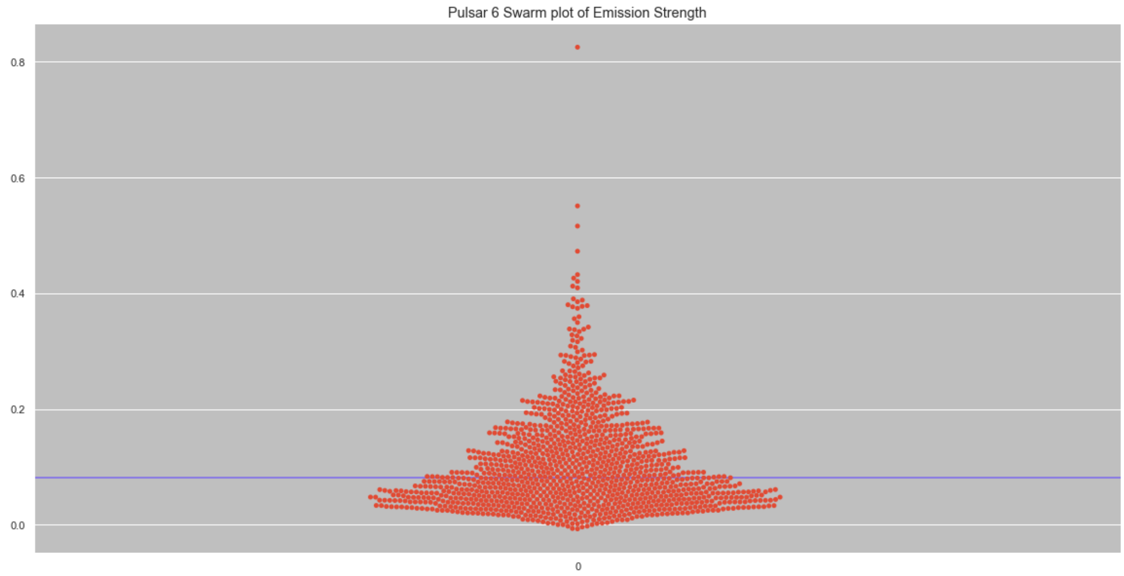
```
C:\Users\tajki\anaconda3\lib\site-packages\matplotlib\collections.py:1003:
RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```




```
[ ]: print(len(pulsar[(pulsar.Brightness > 0.081228)]))
     print(len(pulsar[(pulsar.Brightness < 0.081228)]))
```

```
609
609
```

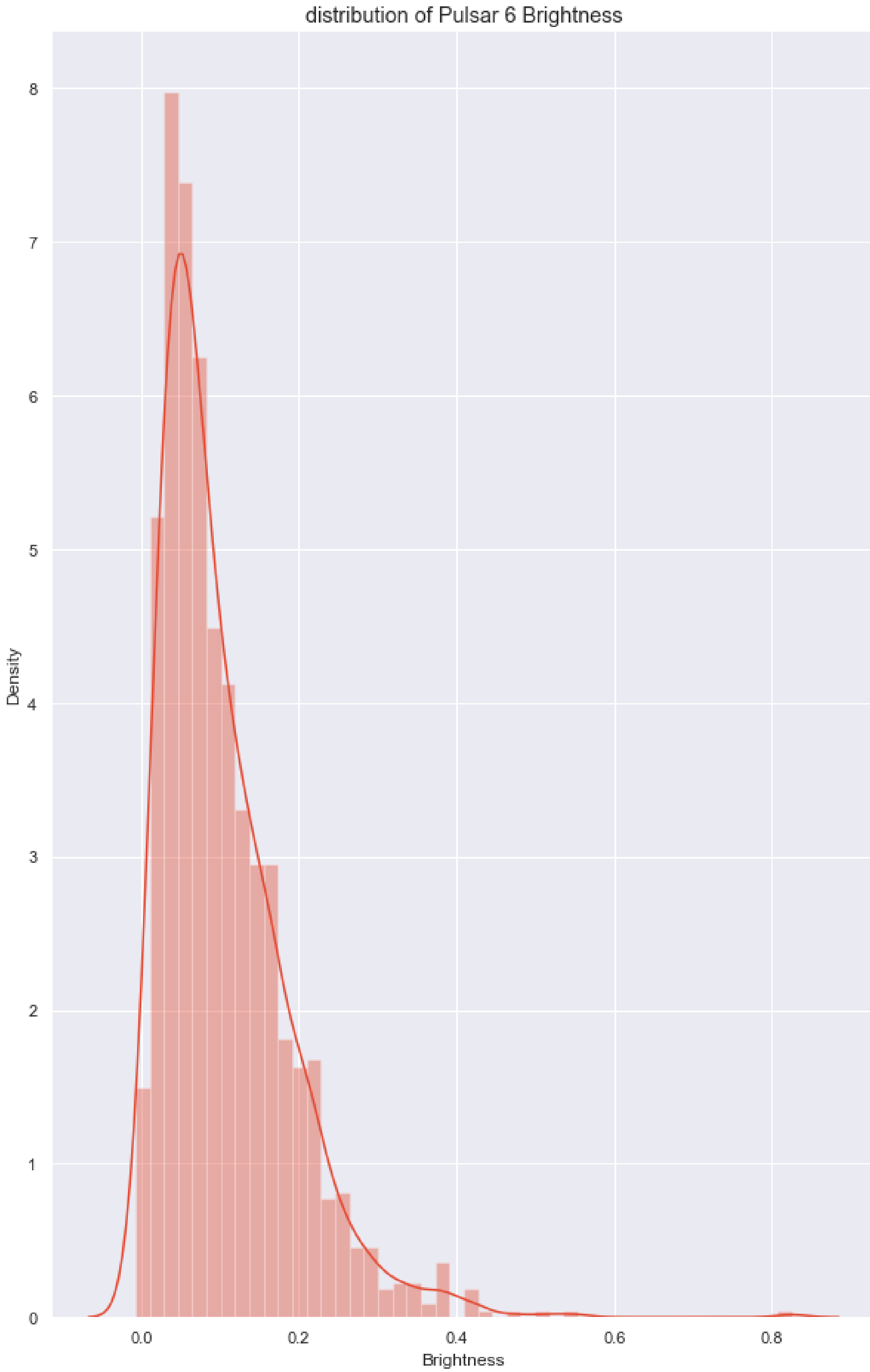
```
[ ]: plt.figure(figsize=(20,10))
     sns.set_style("darkgrid", {"axes.facecolor": ".75"})
     strength = pulsar.Brightness.values
     ax = plt.axhline( y=0.081228, ls='-',c='mediumslateblue')
     ax = sns.swarmplot(data=pulsar["Brightness"], c="blue").set_title('Pulsar 6␣
     ↪Swarm plot of Emission Strength')
```



[ ]:
```
plt.figure(figsize=(10, 16))
with sns.axes_style('darkgrid'):
    sns.distplot(pulsar.Brightness)
plt.title("distribution of Pulsar 6 Brightness")
```

```
C:\Users\tajki\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

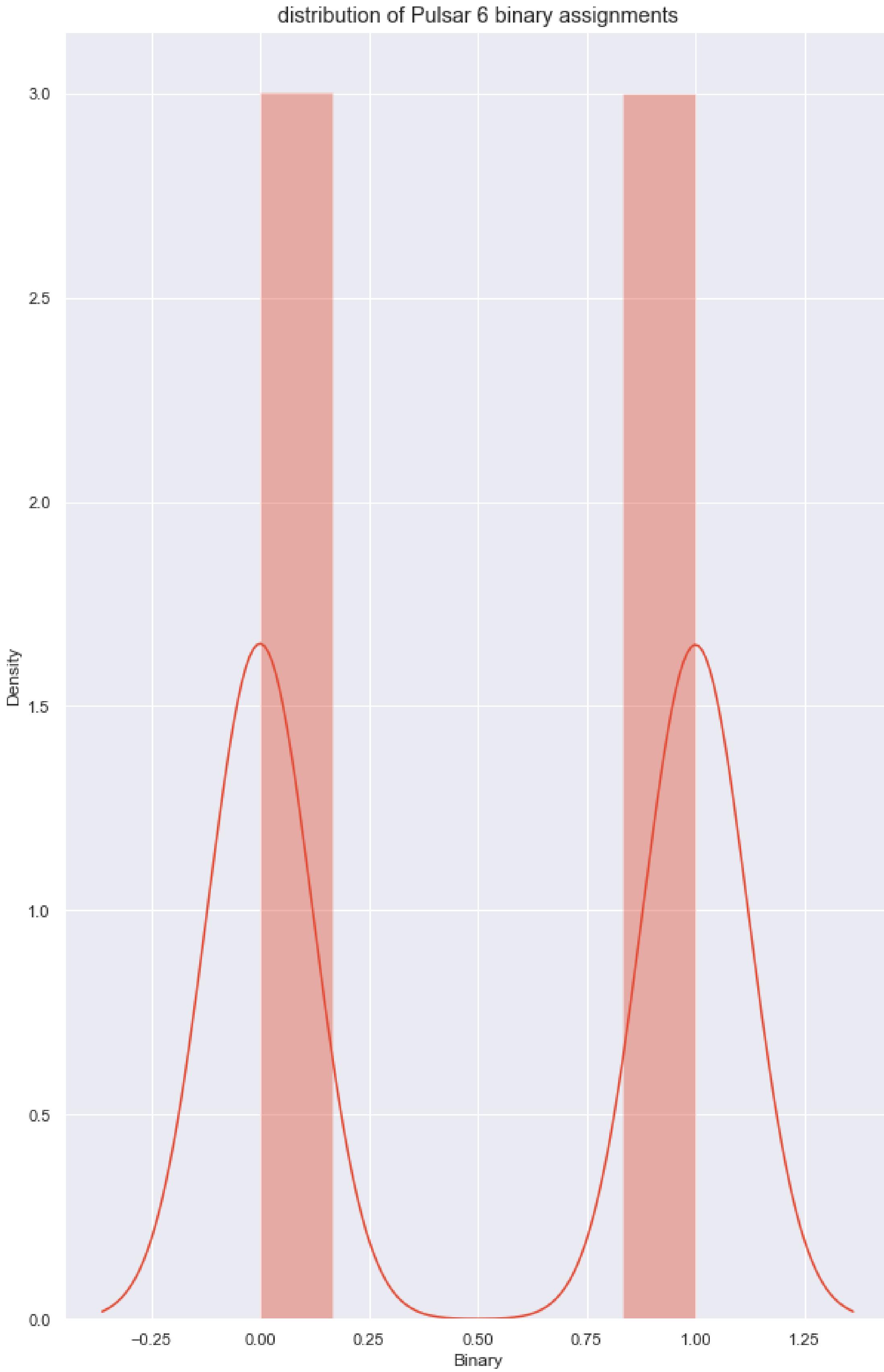
[ ]:
```
Text(0.5, 1.0, 'distribution of Pulsar 6 Brightness')
```


distribution of Pulsar 6 Brightness
Density
Brightness
0
1
2
3
4
5
6
7
8
0.0
0.2
0.4
0.6
0.8

[ ]:
```
plt.figure(figsize=(10, 16))
with sns.axes_style('darkgrid'):
    sns.distplot(pulsar.Binary)
plt.title("distribution of Pulsar 6 binary assignments")
```

```
C:\Users\tajki\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[ ]:
```
Text(0.5, 1.0, 'distribution of Pulsar 6 binary assignments')
```

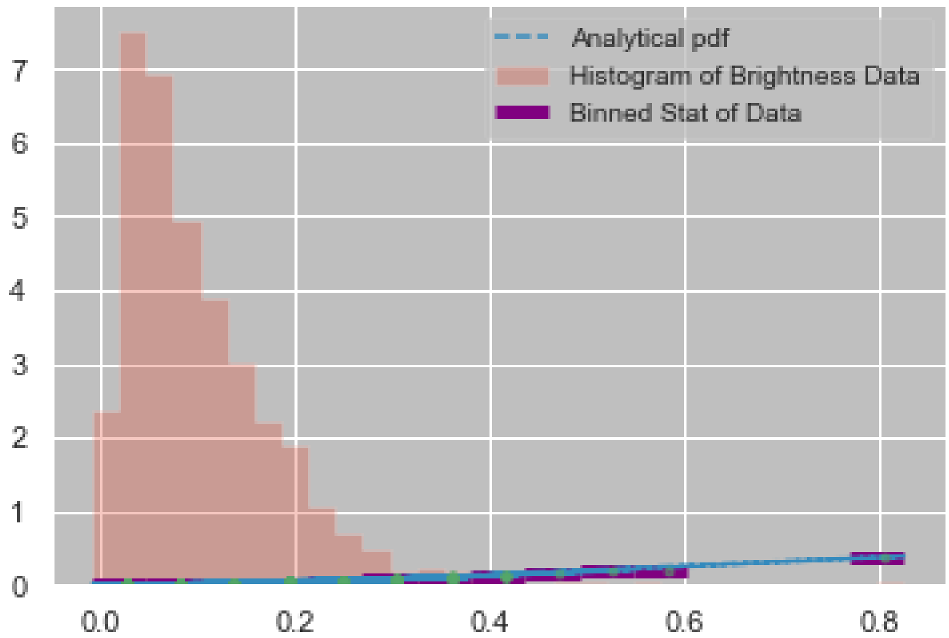

distribution of Pulsar 6 binary assignments
Density
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
Binary

4 Rolling Medians, Rolling Means, Binned Medians and Binned Mean analysis.

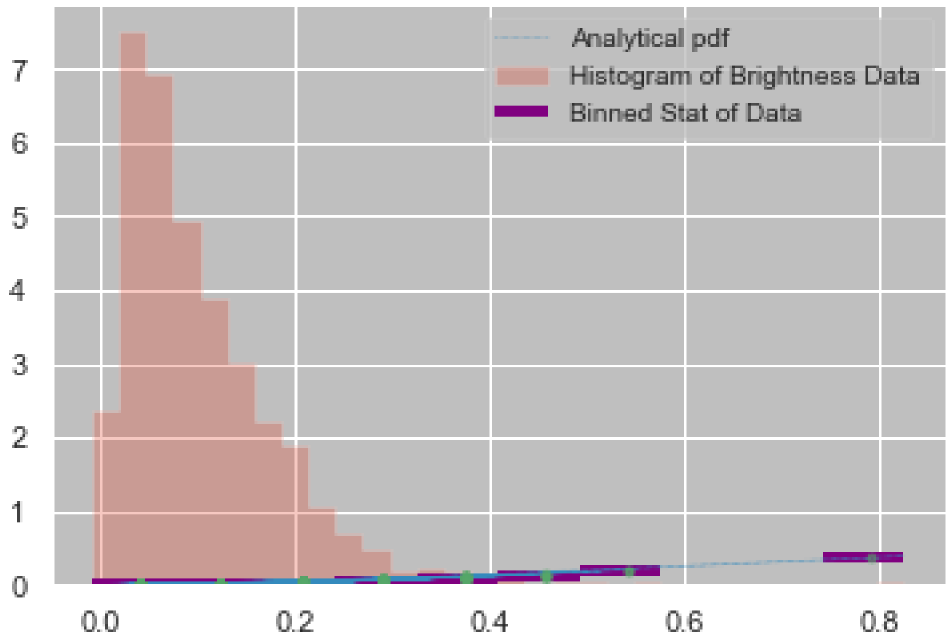
[ ]:
```
data = pulsar["Brightness"]
data
```

[ ]:
```
0       5.390386e-02
1       5.865279e-02
2       1.102083e-01
3       3.471609e-02
4       5.610133e-02
            …
1214    4.321559e-02
1215    1.830750e-02
1216    1.155671e-01
1217    1.562609e-02
1218   -1.137418e-08
Name: Brightness, Length: 1219, dtype: float64
```

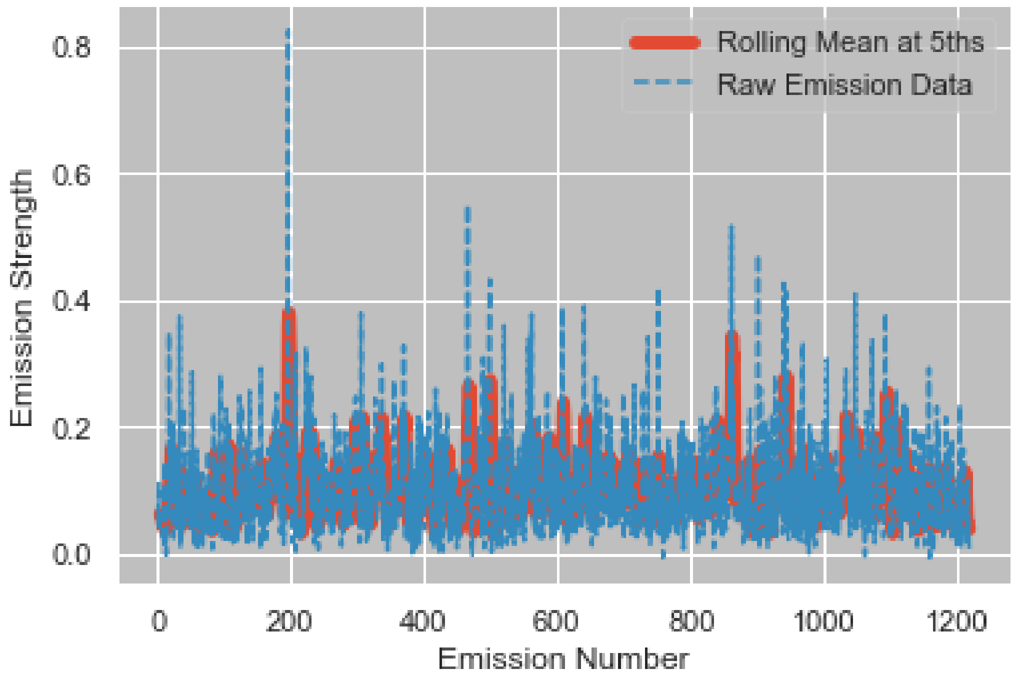
[ ]:
```
dataPDF = stats.maxwell.pdf(data)
bin_means, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='mean', bins=15)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

plt.figure()
plt.hist(data, bins=30, density=True, histtype='stepfilled', alpha=0.3,␣
 ↪label='Histogram of Brightness Data')
plt.plot(data, dataPDF, '--', label = "Analytical pdf")
plt.hlines(bin_means, bin_edges[:-1], bin_edges[1:], colors='purple', lw=5,␣
 ↪label='Binned Stat of Data')
plt.plot((binnumber - 0.5) * bin_width, dataPDF, 'g.', alpha=0.5)
plt.legend(fontsize=10)
plt.show()
```



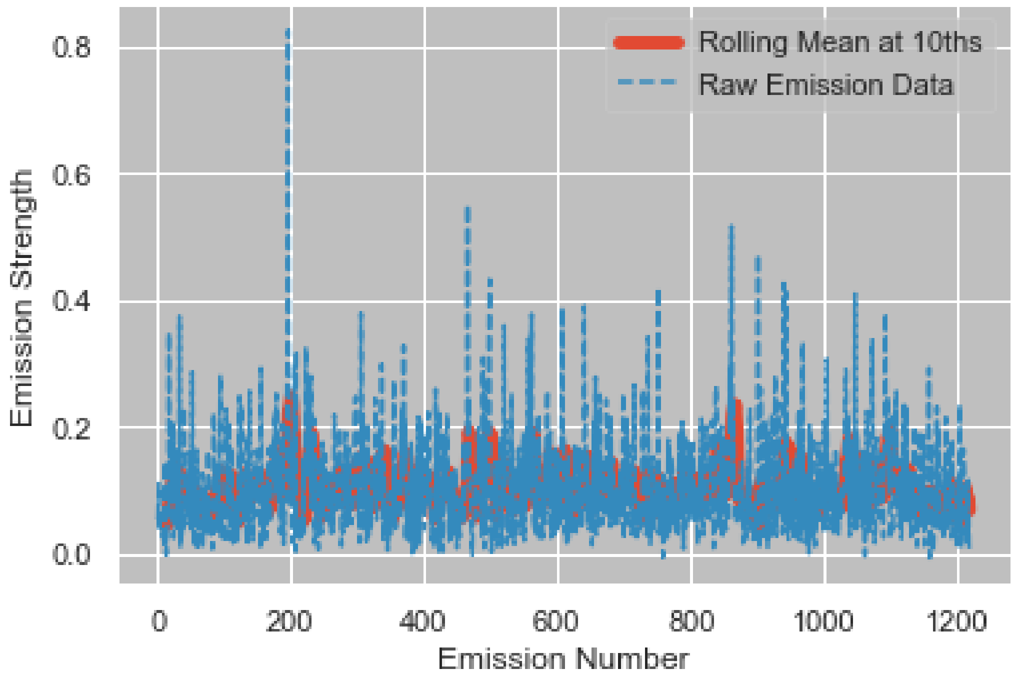
[ ]:
```
bin_median, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='median', bins=10)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

plt.figure()
plt.hist(data, bins=30, density=True, histtype='stepfilled', alpha=0.3,␣
 ↪label='Histogram of Brightness Data')
plt.plot(data, dataPDF, ':', label = "Analytical pdf", lw=0.5)
plt.hlines(bin_median, bin_edges[:-1], bin_edges[1:], colors='purple', lw=4,␣
 ↪label='Binned Stat of Data')
plt.plot((binnumber - 0.5) * bin_width, dataPDF, 'g.', alpha=0.5)
plt.legend(fontsize=10)
plt.show()
```



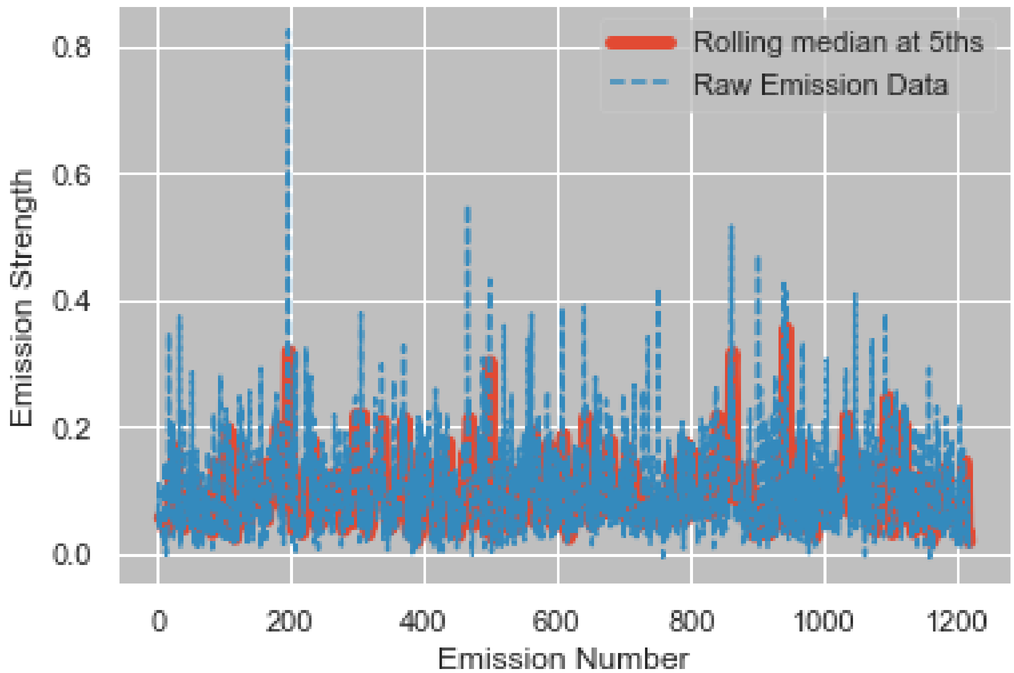
[ ]:
```
pulsar['RollingMeanEmissions5ths'] = pulsar["Brightness"].rolling(5).mean()

plt.plot(pulsar['RollingMeanEmissions5ths'], label="Rolling Mean at 5ths", lw=5)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", linestyle='--')
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```



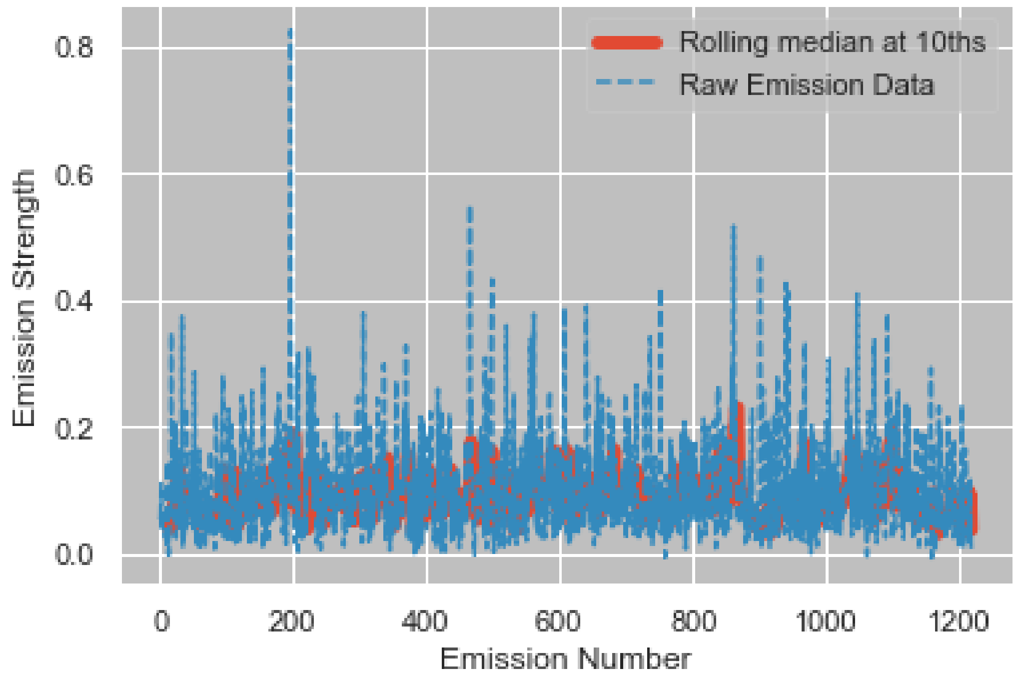
[ ]:
```
pulsar['RollingMeanEmissions10ths'] = pulsar["Brightness"].rolling(10).mean()

plt.plot(pulsar['RollingMeanEmissions10ths'], label="Rolling Mean at 10ths",␣
 ↪lw=5)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", linestyle='--')
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```




```
[ ]: pulsar['RollingMedianEmissions5ths'] = pulsar["Brightness"].rolling(5).median()

     plt.plot(pulsar['RollingMedianEmissions5ths'], label="Rolling median at 5ths",␣
      ↪lw=5)
     plt.plot(pulsar['Brightness'], label= "Raw Emission Data", linestyle='--')
     plt.legend()
     plt.ylabel('Emission Strength')
     plt.xlabel('Emission Number')
     plt.show()
```



[ ]:
```
pulsar['RollingMedianEmissions10ths'] = pulsar["Brightness"].rolling(10).
    ↪median()

plt.plot(pulsar['RollingMedianEmissions10ths'], label="Rolling median at␣
    ↪10ths", lw=5)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", linestyle='--')
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```



[ ]:
```
pulsar.head(25)
```

[ ]:

	Pulse Number	Brightness	Uncertainty	Binary	RollingMeanEmissions5ths	\
0	1	0.053904	0.005560	0	NaN	
1	2	0.058653	0.004821	0	NaN	
2	3	0.110208	0.005196	1	NaN	
3	4	0.034716	0.004729	0	NaN	
4	5	0.056101	0.004619	0	0.062716	
5	6	0.046168	0.005074	0	0.061169	
6	7	0.055648	0.004916	0	0.060568	
7	8	0.060890	0.004581	0	0.050705	
8	9	0.024388	0.004922	0	0.048639	
9	10	0.039370	0.004633	0	0.045293	
10	11	0.009141	0.004581	0	0.037888	
11	12	0.145273	0.005053	1	0.055813	
12	13	0.039953	0.004938	0	0.051625	
13	14	-0.002554	0.004409	0	0.046237	
14	15	0.035696	0.004903	0	0.045502	
15	16	0.046869	0.004706	0	0.053048	
16	17	0.082637	0.004596	1	0.040520	
17	18	0.349419	0.006828	1	0.102413	
18	19	0.058343	0.004650	0	0.114593	
19	20	0.090261	0.005068	1	0.125506	
20	21	0.120429	0.005141	1	0.140218	

```
21             22    0.209730     0.005389      1                  0.165637
22             23    0.088045     0.004945      1                  0.113362
23             24    0.203736     0.008553      1                  0.142440
24             25    0.024098     0.004641      0                  0.129208

    RollingMeanEmissions10ths  RollingMedianEmissions5ths  \
0                         NaN                         NaN
1                         NaN                         NaN
2                         NaN                         NaN
3                         NaN                         NaN
4                         NaN                    0.056101
5                         NaN                    0.056101
6                         NaN                    0.055648
7                         NaN                    0.055648
8                         NaN                    0.055648
9                    0.054005                    0.046168
10                   0.049528                    0.039370
11                   0.058190                    0.039370
12                   0.051165                    0.039370
13                   0.047438                    0.039370
14                   0.045397                    0.035696
15                   0.045468                    0.039953
16                   0.048166                    0.039953
17                   0.077019                    0.046869
18                   0.080415                    0.058343
19                   0.085504                    0.082637
20                   0.096633                    0.090261
21                   0.103078                    0.120429
22                   0.107888                    0.090261
23                   0.128517                    0.120429
24                   0.127357                    0.120429

    RollingMedianEmissions10ths
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN
5                           NaN
6                           NaN
7                           NaN
8                           NaN
9                      0.054776
10                     0.050908
11                     0.050908
12                     0.043061
13                     0.043061
```

```
14                       0.039662
15                       0.039662
16                       0.039662
17                       0.039662
18                       0.043411
19                       0.052606
20                       0.070490
21                       0.070490
22                       0.085341
23                       0.089153
24                       0.089153
```

4.1 Binary Classification

[ ]:
```
X = pulsar[['Brightness', 'Uncertainty']]
y = pulsar['Binary']
```

[ ]:
```
X.head()
```

[ ]:
```
   Brightness  Uncertainty
0    0.053904     0.005560
1    0.058653     0.004821
2    0.110208     0.005196
3    0.034716     0.004729
4    0.056101     0.004619
```

[ ]:
```
y.head()
```

[ ]:
```
0    0
1    0
2    1
3    0
4    0
Name: Binary, dtype: int32
```

[ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y , test_size=0.20)
```

[ ]:
```
from sklearn.preprocessing import StandardScaler

train_scaler = StandardScaler()
X_train = train_scaler.fit_transform(X_train)

test_scaler = StandardScaler()
X_test = test_scaler.fit_transform(X_test)
```

[ ]:
```
model = LogisticRegression()

model.fit(X_train, y_train)
```

[ ]:
```
LogisticRegression()
```

[ ]:
```
predictions = model.predict(X_test)
```

[ ]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, predictions)

TN, FP, FN, TP = confusion_matrix(y_test, predictions).ravel()

print('True Positive(TP)  = ', TP)
print('False Positive(FP) = ', FP)
print('True Negative(TN)  = ', TN)
print('False Negative(FN) = ', FN)
```

```
True Positive(TP)  =  109
False Positive(FP) =  6
True Negative(TN)  =  129
False Negative(FN) =  0
```

[ ]:
```
accuracy =  (TP + TN) / (TP + FP + TN + FN)

print("Accuracy of the model is ", accuracy)
```

```
Accuracy of the model is  0.9754098360655737
```

4.2 Bidirectional LSTM Model

[ ]:
```
# making a list with the brightness and uncertainty values
values_list = pulsar[['Brightness', 'Uncertainty']].values.tolist()
values_list[:10]
```

[ ]:
```
[[0.05390386, 0.005559608],
 [0.05865279, 0.004821059],
 [0.1102083, 0.005195663],
 [0.03471609, 0.004728795],
 [0.05610133, 0.00461856],
 [0.04616798, 0.005074086],
 [0.05564797, 0.004915987],
 [0.06089036, 0.004580757],
 [0.02438825, 0.00492155],
 [0.0393704, 0.004633388]]
```

[ ]:
```
from sklearn import preprocessing

# normalizing the values
values_list = preprocessing.normalize(values_list)
```

[ ]:
```
# function for spliting a list in a format we can use in the model
def split_list(blist, steps):
    X, y = list(), list()
    for i in range(len(blist)):
        end_ix = i + steps
        if end_ix > len(blist)-1:
            break
        list_x, list_y = blist[i:end_ix], blist[end_ix][0]
        X.append(list_x)
        y.append(list_y)
    return array(X), array(y)
```

[ ]:
```
# splitting the list
X, y = split_list(values_list, 100)

# reshaping the list to feed the model
X = X.reshape((X.shape[0], X.shape[1], 2))
```

[ ]:
```
# splitting the list into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size=0.20)
```

[ ]:
```
X_train.shape
```

[ ]:
```
(895, 100, 2)
```

[ ]:
```
# setting the parameters for the lstm model and compiling it
model = Sequential()
model.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(100, 2)))
model.add(Dense(25, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(6, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
# training the model
history = model.fit(X_train, y_train, epochs=50, verbose=1,␣
 ↪batch_size=(int(X_train.shape[0]/50)))
```

```
Epoch 1/50
53/53 [==============================] - 3s 28ms/step - loss: 0.6741 - accuracy:
0.0000e+00
Epoch 2/50
```

```
53/53 [==============================] - 2s 30ms/step - loss: 0.4975 - accuracy:
0.0000e+00
Epoch 3/50
53/53 [==============================] - 2s 39ms/step - loss: 0.4426 - accuracy:
0.0000e+00
Epoch 4/50
53/53 [==============================] - 2s 38ms/step - loss: 0.3076 - accuracy:
0.0000e+00
Epoch 5/50
53/53 [==============================] - 2s 44ms/step - loss: 0.2176 - accuracy:
0.0000e+00
Epoch 6/50
53/53 [==============================] - 2s 42ms/step - loss: 0.1699 - accuracy:
0.0000e+00
Epoch 7/50
53/53 [==============================] - 2s 46ms/step - loss: 0.1271 - accuracy:
0.0000e+00
Epoch 8/50
53/53 [==============================] - 2s 43ms/step - loss: 0.1077 - accuracy:
0.0000e+00
Epoch 9/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0990 - accuracy:
0.0000e+00
Epoch 10/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0955 - accuracy:
0.0000e+00
Epoch 11/50
53/53 [==============================] - 2s 43ms/step - loss: 0.0939 - accuracy:
0.0000e+00
Epoch 12/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0934 - accuracy:
0.0000e+00
Epoch 13/50
53/53 [==============================] - 2s 46ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 14/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 15/50
53/53 [==============================] - 2s 47ms/step - loss: 0.0934 - accuracy:
0.0000e+00
Epoch 16/50
53/53 [==============================] - 2s 43ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 17/50
53/53 [==============================] - 2s 37ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 18/50
```

```
53/53 [==============================] - 2s 43ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 19/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 20/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 21/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 22/50
53/53 [==============================] - 2s 47ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 23/50
53/53 [==============================] - 2s 42ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 24/50
53/53 [==============================] - 2s 39ms/step - loss: 0.0931 - accuracy:
0.0000e+00
Epoch 25/50
53/53 [==============================] - 2s 40ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 26/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0934 - accuracy:
0.0000e+00
Epoch 27/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 28/50
53/53 [==============================] - 2s 42ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 29/50
53/53 [==============================] - 2s 47ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 30/50
53/53 [==============================] - 2s 43ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 31/50
53/53 [==============================] - 2s 42ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 32/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 33/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 34/50
```

```
53/53 [==============================] - 2s 42ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 35/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 36/50
53/53 [==============================] - 2s 46ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 37/50
53/53 [==============================] - 3s 47ms/step - loss: 0.0931 - accuracy:
0.0000e+00
Epoch 38/50
53/53 [==============================] - 3s 49ms/step - loss: 0.0933 - accuracy:
0.0000e+00
Epoch 39/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0935 - accuracy:
0.0000e+00
Epoch 40/50
53/53 [==============================] - 2s 47ms/step - loss: 0.0931 - accuracy:
0.0000e+00
Epoch 41/50
53/53 [==============================] - 2s 40ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 42/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 43/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0931 - accuracy:
0.0000e+00
Epoch 44/50
53/53 [==============================] - 3s 49ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 45/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 46/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0930 - accuracy:
0.0000e+00
Epoch 47/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0932 - accuracy:
0.0000e+00
Epoch 48/50
53/53 [==============================] - 2s 46ms/step - loss: 0.0930 - accuracy:
0.0000e+00
Epoch 49/50
53/53 [==============================] - 2s 43ms/step - loss: 0.0931 - accuracy:
0.0000e+00
Epoch 50/50
```

```
53/53 [==============================] - 2s 40ms/step - loss: 0.0930 - accuracy:
0.0000e+00
```

```
[ ]: # predicting the y/brightness values for the test set
y_pred = model.predict(X_test, verbose=0)
y_pred[:10]
```

```
[ ]: array([[0.9813596 ],
       [0.9805201 ],
       [0.98130524],
       [0.97970897],
       [0.9797267 ],
       [0.98002166],
       [0.9799235 ],
       [0.9785363 ],
       [0.9804213 ],
       [0.97794783]], dtype=float32)
```

```
[ ]: # evaluating the model
model.evaluate(X_test, y_test)
```

```
7/7 [==============================] - 1s 12ms/step - loss: 0.0935 - accuracy:
0.0000e+00
```

```
[ ]: [0.0934809073805809, 0.0]
```

4.3 ML Evaluation.

4.3.1 Logistic Regression

Rewards no significant results for this type of analysis and is dropped for a LSTM attempt

4.3.2 Bidirectional LSTM

Loss is low so the model is performing well. But the accuracy is low therefore unable to obtain trend and therefore not rewarding any information. This means we cannot predict any of the values with confidence.

5 Preliminary runs test

5.0.1 Math Logic

$$Z = \frac{R - \tilde{R}}{s_R}$$

$$\tilde{R} = \frac{2_{n1n2}}{n1 + n2} + 1$$

$$s_R^2 = \frac{2_{n1n2}(2n1n2 - n1 - n2)}{(n1 + n2)^2(n1 + n2 - 1)}$$

link to resource: https://www.geeksforgeeks.org/runs-test-of-randomness-in-python/

$ Z_{critical} = 1.96 $ as the confidence interval level of 95% thus this is a 2 tailed test. If the probability as corrosponding to this confidence interval $ H_{null} $ will be rejected as it is not statistically significant as denoted by $|Z| > Z_{critical} $

There is also code attempting to change it from a z-score probability to a P-score for ease of understanding and clarity.

6 FUNCTION CODE FOR RUNS TEST

[ ]:
```
binaryData1 = pulsar['Binary'].tolist()
print("pulsar6 original: ",binaryData1)
```

```
pulsar6 original:  [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0,
0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0,
0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
```

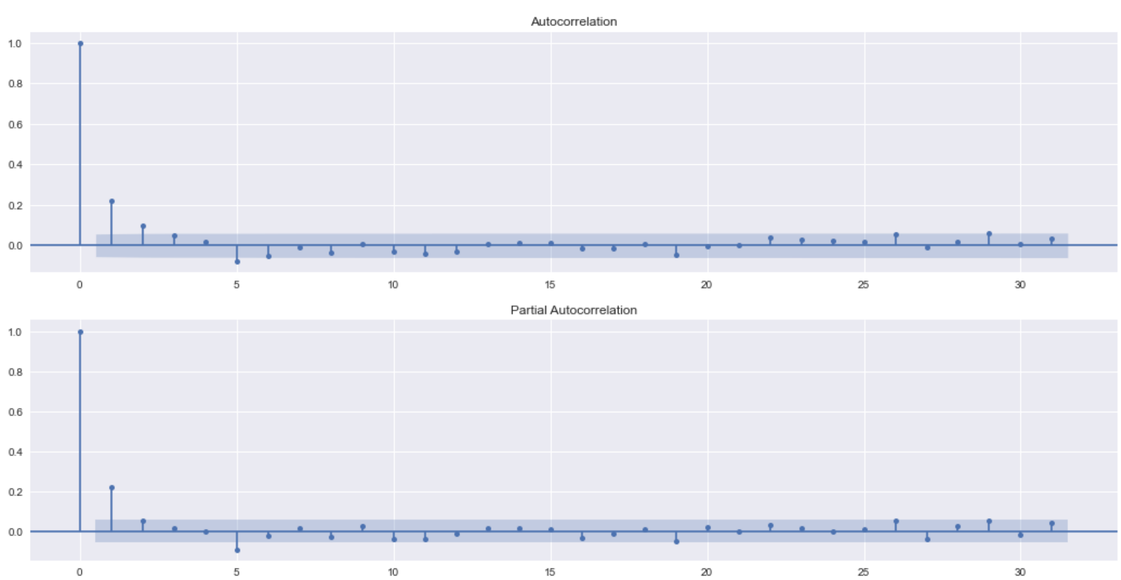
```
1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,
1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0]
```

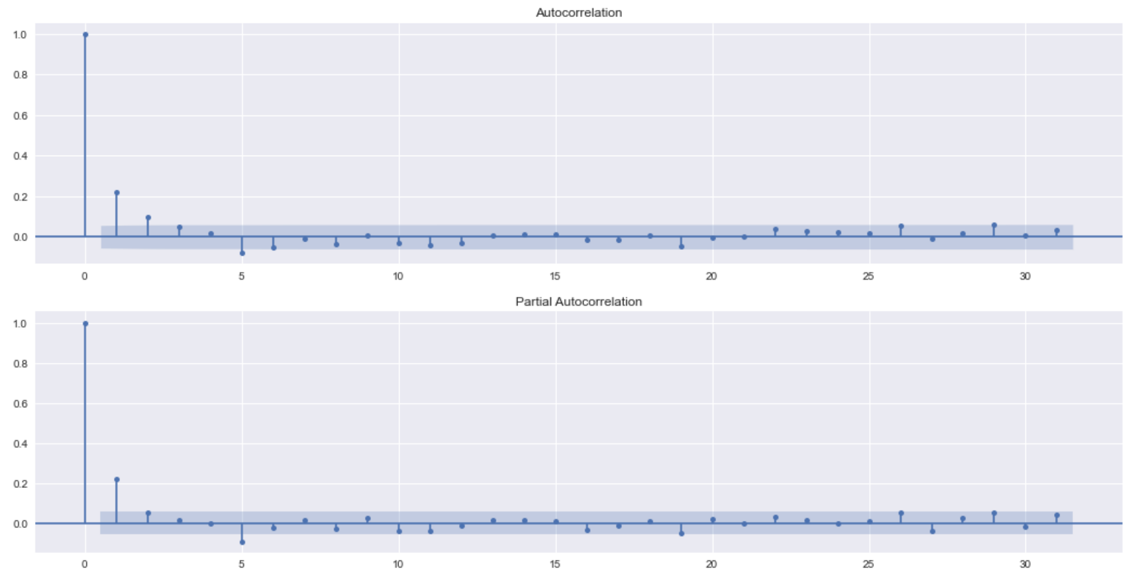
7 Below we begin autocorrelation and autocovariance analysis

To get started with this I am playing around with guide from: https://towardsdatascience.com/a-step-by-step-guide-to-calculating-autocorrelation-and-partial-autocorrelation-8c4342b784e8

[ ]:
```
plt.style.use("seaborn")
plt.rcParams["figure.figsize"] = (18, 9)

fig, ax = plt.subplots(2,1)

plot_acf(pulsar['Brightness'], ax=ax[0])
plot_pacf(pulsar['Brightness'], ax=ax[1], method="ols")
```

[ ]:





[ ]:
```
acf(pulsar['Brightness'], nlags=10)
```

```
C:\Users\tajki\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:667:
FutureWarning: fft=True will become the default after the release of the 0.12
release of statsmodels. To suppress this warning, explicitly set fft=False.
  warnings.warn(
```

[ ]:
```
array([ 1.        ,  0.22161581,  0.09940415,  0.04669096,  0.01876941,
       -0.07818365, -0.0539178 , -0.01220963, -0.03566829,  0.00520742,
       -0.03014362])
```

[ ]:
```
acfpulsar = pd.DataFrame()
for lag in range(0,11):
    acfpulsar[f"B_lag_{lag}"] = pulsar['Brightness'].shift(lag)


acfpulsar
```

[ ]:

	B_lag_0	B_lag_1	B_lag_2	B_lag_3	B_lag_4	B_lag_5	\
0	5.390386e-02	NaN	NaN	NaN	NaN	NaN	
1	5.865279e-02	0.053904	NaN	NaN	NaN	NaN	
2	1.102083e-01	0.058653	0.053904	NaN	NaN	NaN	
3	3.471609e-02	0.110208	0.058653	0.053904	NaN	NaN	
4	5.610133e-02	0.034716	0.110208	0.058653	0.053904	NaN	
…	…	…	…	…	…	…	
1214	4.321559e-02	0.031916	0.030713	0.116777	0.144606	0.165039	
1215	1.830750e-02	0.043216	0.031916	0.030713	0.116777	0.144606	
1216	1.155671e-01	0.018308	0.043216	0.031916	0.030713	0.116777	

```
1217  1.562609e-02  0.115567  0.018308  0.043216  0.031916  0.030713
1218 -1.137418e-08  0.015626  0.115567  0.018308  0.043216  0.031916

       B_lag_6   B_lag_7   B_lag_8   B_lag_9  B_lag_10
0          NaN       NaN       NaN       NaN       NaN
1          NaN       NaN       NaN       NaN       NaN
2          NaN       NaN       NaN       NaN       NaN
3          NaN       NaN       NaN       NaN       NaN
4          NaN       NaN       NaN       NaN       NaN
…          …        …        …        …        …
1214  0.148642  0.071752  0.008108  0.038793  0.084002
1215  0.165039  0.148642  0.071752  0.008108  0.038793
1216  0.144606  0.165039  0.148642  0.071752  0.008108
1217  0.116777  0.144606  0.165039  0.148642  0.071752
1218  0.030713  0.116777  0.144606  0.165039  0.148642

[1219 rows x 11 columns]
```

[ ]:
```
acfpulsar.corr()["B_lag_0"].values
```

[ ]:
```
array([ 1.        ,  0.22179701,  0.09954441,  0.04675654,  0.01880625,
       -0.07839106, -0.05409556, -0.01226841, -0.03581717,  0.00521062,
       -0.03030331])
```

7.0.1 Getting every 5th as per the auto correlation

7.0.2 Creating a new set of discrete 100 sets and examining them specifically

7.0.3 Further Random testing to move into extensive testing

Getting every 5th as per the auto correlation

[ ]:
```
held5ths = pulsar[pulsar.index % 5 == 0]
held5ths
```

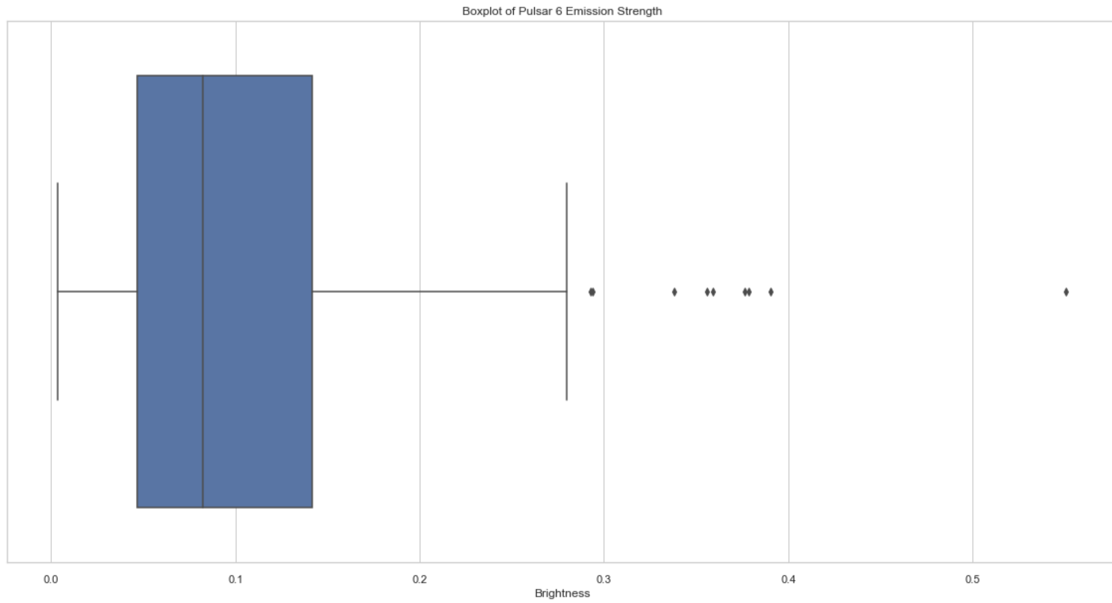
[ ]:
```
      Pulse Number  Brightness  Uncertainty  Binary  RollingMeanEmissions5ths  \
0                1    0.053904     0.005560       0                       NaN
5                6    0.046168     0.005074       0                  0.061169
10              11    0.009141     0.004581       0                  0.037888
15              16    0.046869     0.004706       0                  0.053048
20              21    0.120429     0.005141       1                  0.140218
…              …          …           …     …                       …
1195          1196    0.049626     0.004631       0                  0.072716
1200          1201    0.117575     0.005117       1                  0.055436
1205          1206    0.038793     0.004621       0                  0.079521
1210          1211    0.144606     0.005046       1                  0.107629
1215          1216    0.018308     0.004578       0                  0.048186

      RollingMeanEmissions10ths  RollingMedianEmissions5ths  \
```

```
0                           NaN                           NaN
5                           NaN                      0.056101
10                     0.049528                      0.039370
15                     0.045468                      0.039953
20                     0.096633                      0.090261
…                           …                        …
1195                   0.079188                      0.049626
1200                   0.064076                      0.050588
1205                   0.067478                      0.038793
1210                   0.093575                      0.144606
1215                   0.077908                      0.031916

      RollingMedianEmissions10ths
0                             NaN
5                             NaN
10                       0.050908
15                       0.039662
20                       0.070490
…                            …
1195                     0.056310
1200                     0.050107
1205                     0.044691
1210                     0.077877
1215                     0.057484

[244 rows x 8 columns]
```

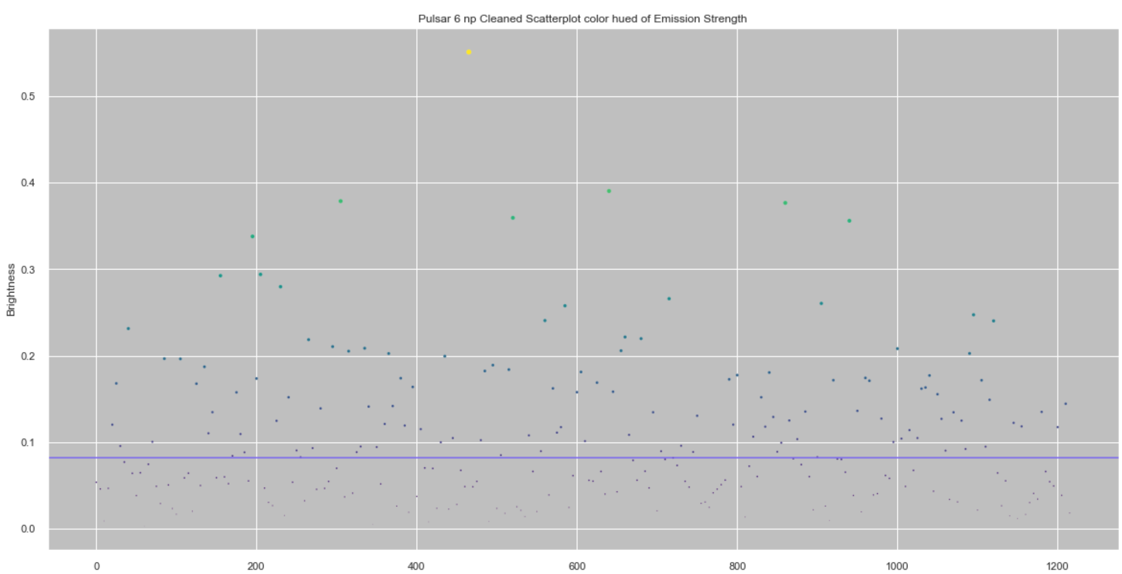
[ ]:
```
medianheld5ths = held5ths["Brightness"].median()
medianheld5ths
```

[ ]:
```
0.08254402
```

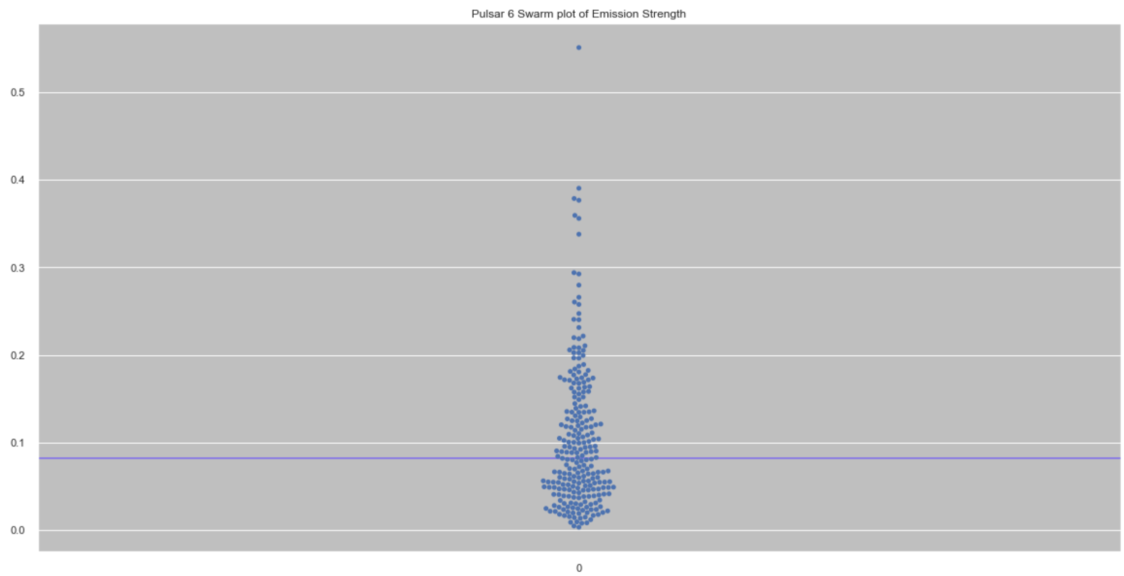
[ ]:
```
plt.figure(figsize=(20,10))
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=held5ths["Brightness"]).set_title("Boxplot of Pulsar 6␣
↪Emission Strength")
```




```
[ ]: plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = held5ths.Brightness.values
ax = sns.scatterplot(data=held5ths["Brightness"], s= strength*50, c=strength,␣
 ↪cmap="viridis", marker="o").set_title('Pulsar 6 np Cleaned Scatterplot color␣
 ↪hued of Emission Strength')
ax = plt.axhline( y=0.08254402, ls='-',c='mediumslateblue')
```



[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = held5ths.Brightness.values
ax = plt.axhline( y=0.08254402, ls='-',c='mediumslateblue')
ax = sns.swarmplot(data=held5ths["Brightness"], c="blue").set_title('Pulsar 6␣
↪Swarm plot of Emission Strength')
```



[ ]:
```
print(len(held5ths[(held5ths.Brightness > 0.08254402)]))
print(len(held5ths[(held5ths.Brightness < 0.08254402)]))
```

```
122
122
```

Randomness testing

[ ]:
```
np.savetxt(r'every5thbinarypulsar5.txt', held5ths.Binary, fmt='%d',␣
↪delimiter='')
np.savetxt(r'allpulsar5.txt', pulsar.Binary, fmt='%d', delimiter='')
```

[ ]:
```
pulsar.Binary
```

[ ]:
```
0       0
1       0
2       1
3       0
4       0
       ..
1214    0
1215    0
```

```
1216    1
1217    0
1218    0
Name: Binary, Length: 1219, dtype: int32
```