pulsar4

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/J1243-6423.pulses", sep = ' ', header = None, names␣
↪= colnames)
```

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

[ ]:
```
(1819, 3)
```

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

[ ]:

	Pulse Number	Brightness	Uncertainty
0	1	0.101127	0.001893
1	2	0.012166	0.001814
2	3	0.021918	0.001835
3	4	0.181179	0.002183
4	5	0.000240	0.001725
5	6	0.085866	0.001723
6	7	0.067280	0.001778
7	8	0.092884	0.002438
8	9	0.083350	0.002101
9	10	0.087871	0.001941
10	11	0.123529	0.002026
11	12	0.097413	0.001878
12	13	0.100649	0.001820
13	14	0.058025	0.001724
14	15	0.116164	0.001948
15	16	0.029203	0.001918
16	17	0.174895	0.002131
17	18	0.200468	0.002571
18	19	0.123890	0.001805

```
19              20     0.083496      0.001856
20              21     0.042757      0.001891
21              22     0.119953      0.001744
22              23     0.096266      0.001911
23              24     0.040698      0.001975
24              25     0.175852      0.002251
```

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

[ ]:
```
       Pulse Number   Brightness  Uncertainty
count   1819.000000  1819.000000  1819.000000
mean     910.000000     0.075070     0.001958
std      525.244388     0.057006     0.000306
min        1.000000    -0.004643     0.001532
25%      455.500000     0.019738     0.001774
50%      910.000000     0.076660     0.001872
75%     1364.500000     0.112285     0.002041
max     1819.000000     0.269903     0.005952
```

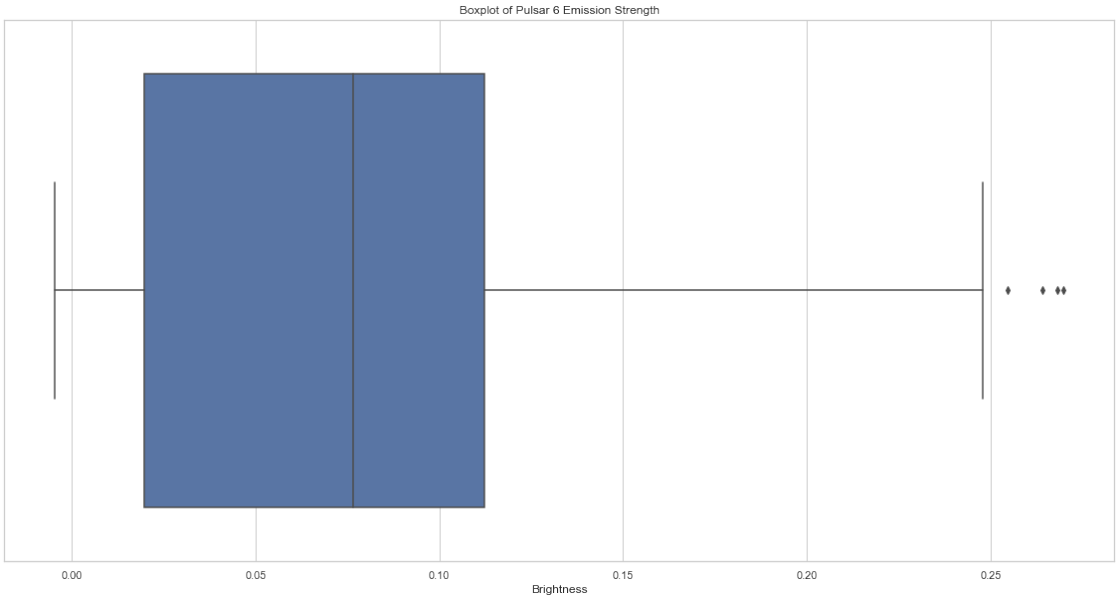
[ ]:
```
nullBoolBrightness = pd.isnull(pulsar["Brightness"])

pulsar[nullBoolBrightness]
```

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

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

[ ]:
```
count    1819.000000
mean        0.075070
std         0.057006
min        -0.004643
25%         0.019738
50%         0.076660
75%         0.112285
max         0.269903
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:  0.07665979
```

[ ]:
```
pulsar
```

[ ]:

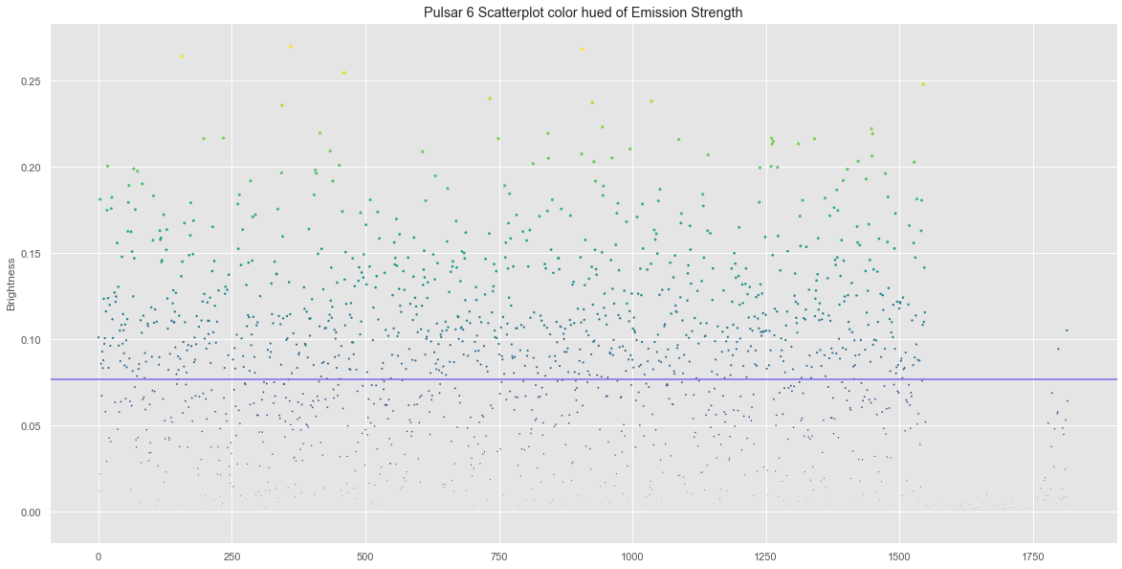
	Pulse Number	Brightness	Uncertainty	Binary
0	1	0.101127	0.001893	1
1	2	0.012166	0.001814	0
2	3	0.021918	0.001835	0
3	4	0.181179	0.002183	1
4	5	0.000240	0.001725	0
…	…	…	…	…
1814	1815	0.105178	0.002086	1
1815	1816	0.064272	0.001995	0
1816	1817	0.000171	0.001730	0
1817	1818	-0.000924	0.001706	0
1818	1819	0.000001	0.001532	0

[1819 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=0.07665979, ls='-',c='mediumslateblue')
```

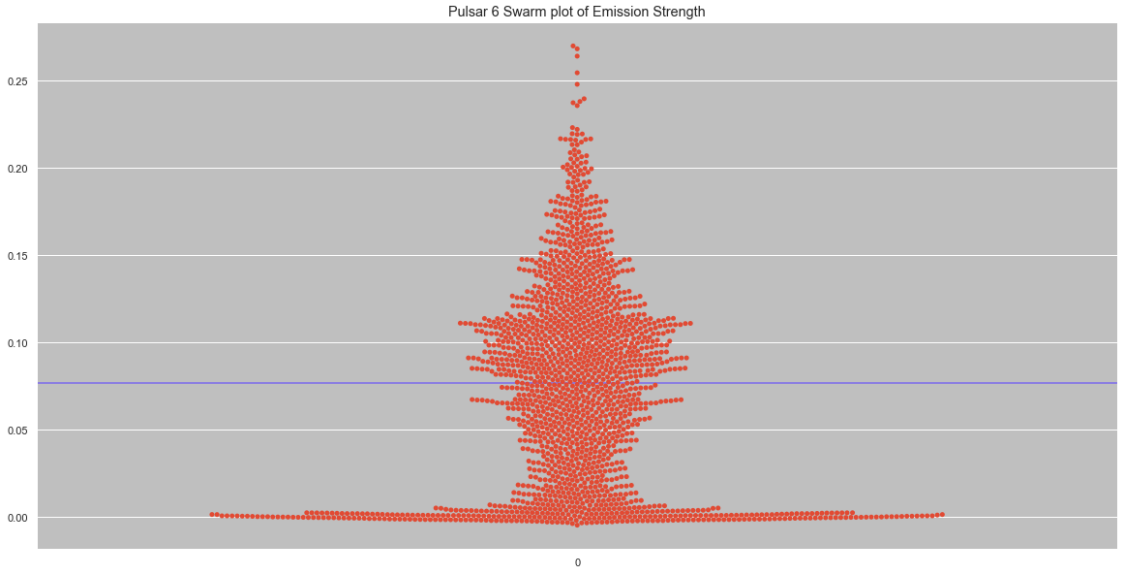
```
C:\Users\tajki\anaconda3\lib\site-packages\matplotlib\collections.py:1003:
RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```



[ ]:
```
print(len(pulsar[(pulsar.Brightness > 0.07665979)]))
print(len(pulsar[(pulsar.Brightness < 0.07665979)]))
```

```
909
909
```

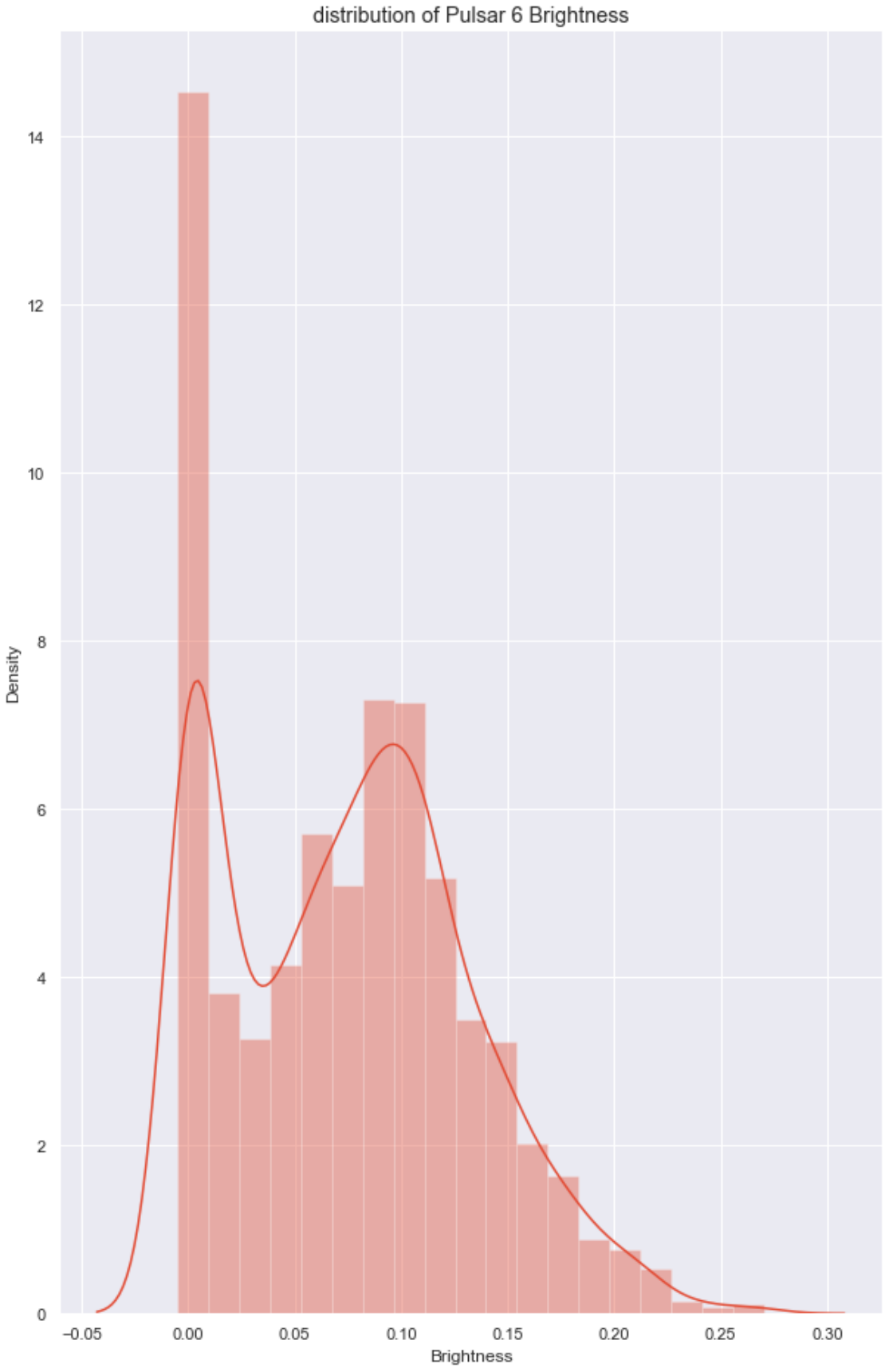
[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = pulsar.Brightness.values
ax = plt.axhline( y=0.07665979, ls='-',c='mediumslateblue')
ax = sns.swarmplot(data=pulsar["Brightness"], c="blue").set_title('Pulsar 6␣
↪Swarm plot of Emission Strength')
```



[ ]:
```
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)
```

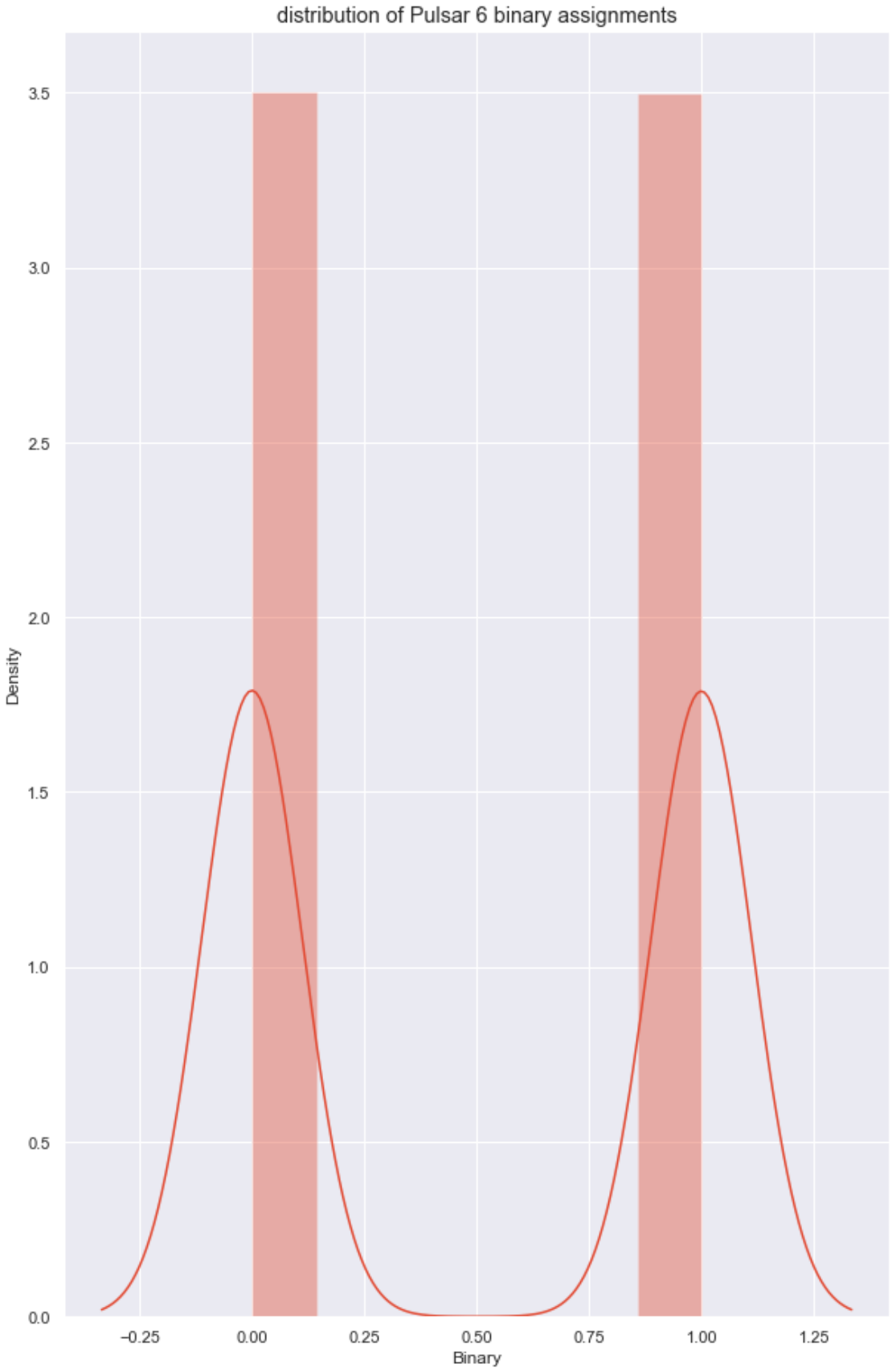
[ ]:
```
Text(0.5, 1.0, 'distribution of Pulsar 6 Brightness')
```


distribution of Pulsar 6 Brightness
Density
Brightness
0
2
4
6
8
10
12
14
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30

[ ]:
```
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')
```

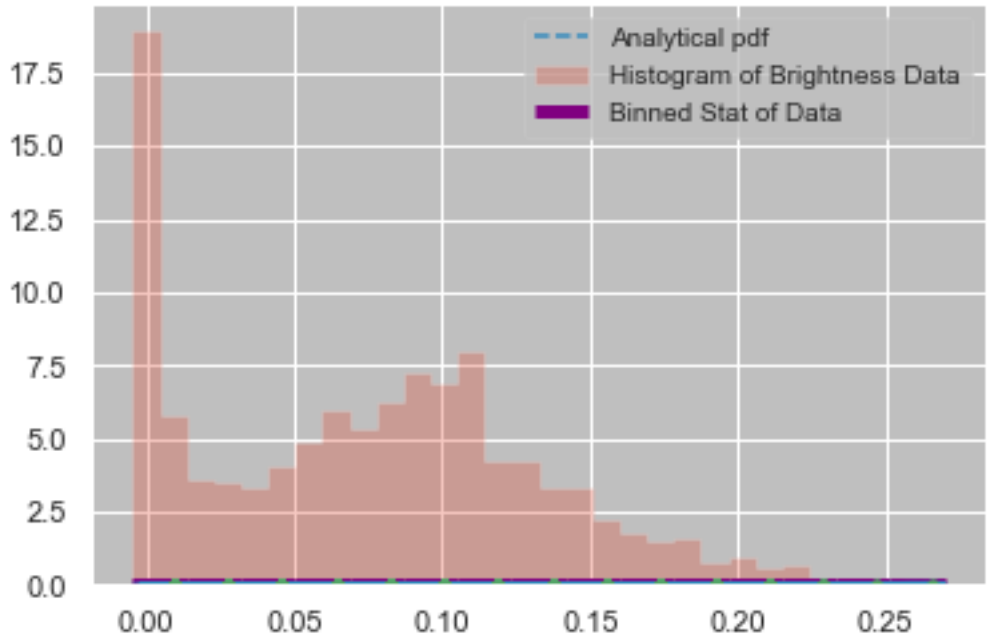

distribution of Pulsar 6 binary assignments
Density
Binary
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
−0.25
0.00
0.25
0.50
0.75
1.00
1.25

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

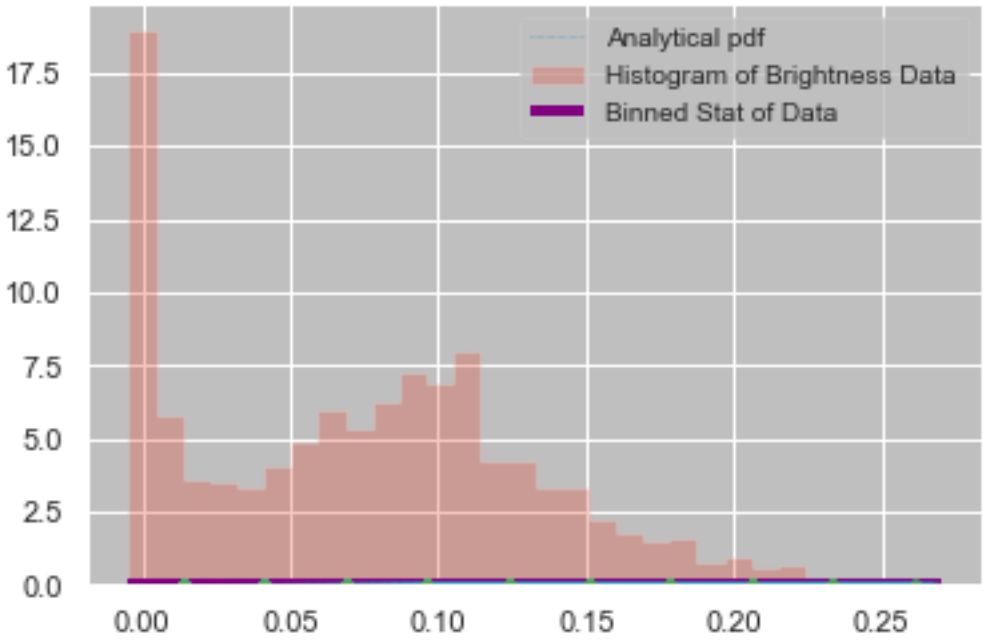
[ ]:
```
data = pulsar["Brightness"]
data
```

[ ]:
```
0       0.101127
1       0.012166
2       0.021918
3       0.181179
4       0.000240
          …
1814    0.105178
1815    0.064272
1816    0.000171
1817   -0.000924
1818    0.000001
Name: Brightness, Length: 1819, dtype: float64
```

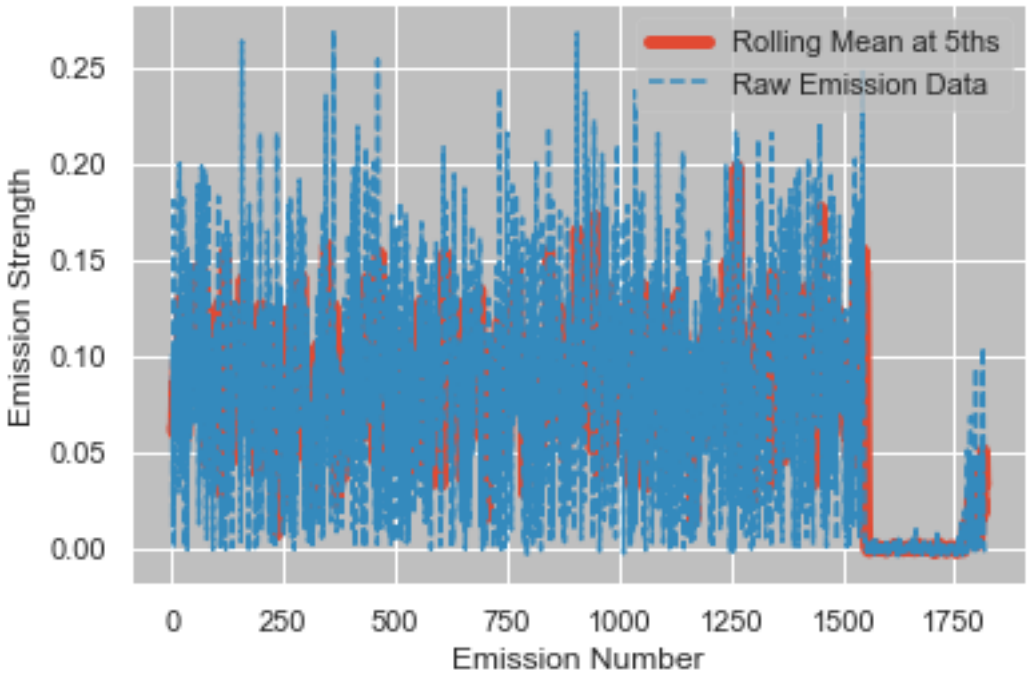
[ ]:
```
dataPDF = stats.maxwell.pdf(data)
bin_means, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='mean', bins=15)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

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()
```



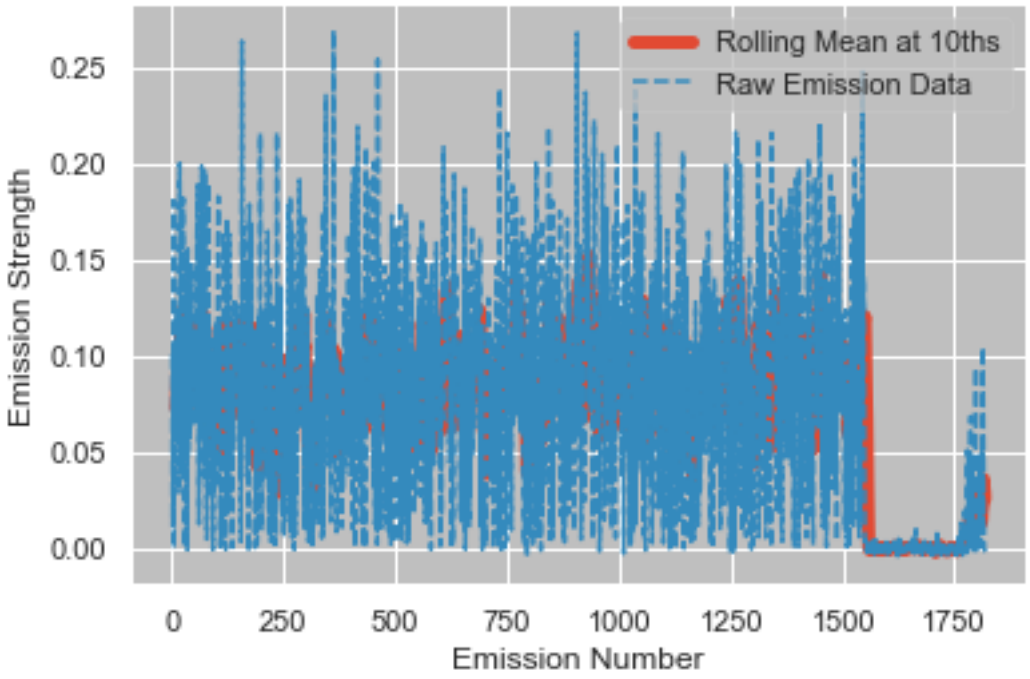
[ ]:
```
bin_median, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='median', bins=10)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

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()
```



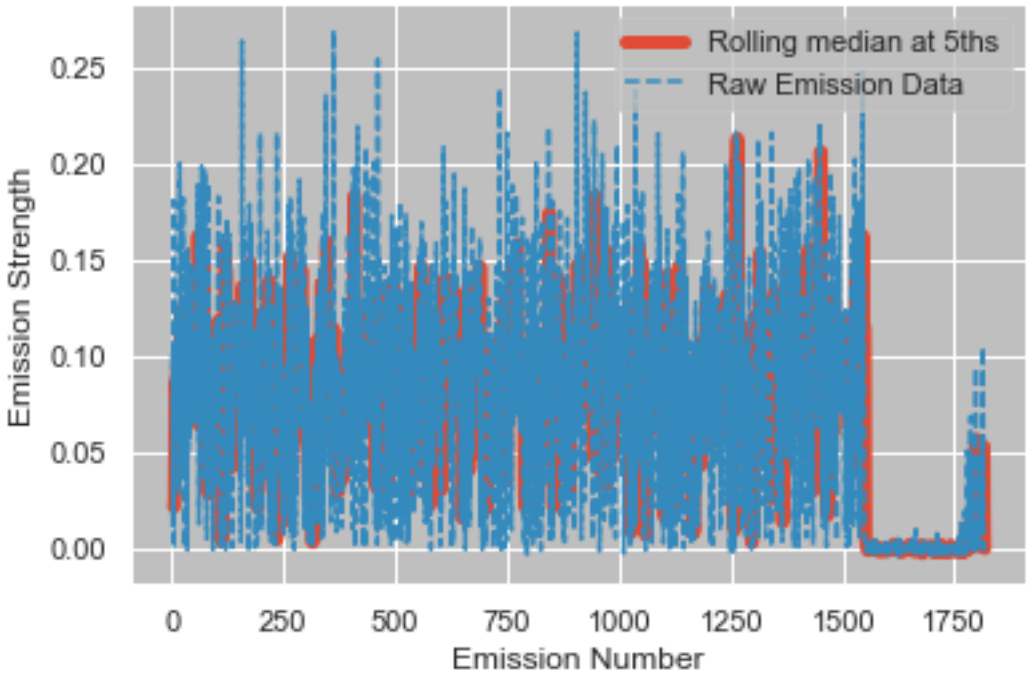
[ ]:
```
pulsar['RollingMeanEmissions5ths'] = pulsar["Brightness"].rolling(5).mean()

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()
```



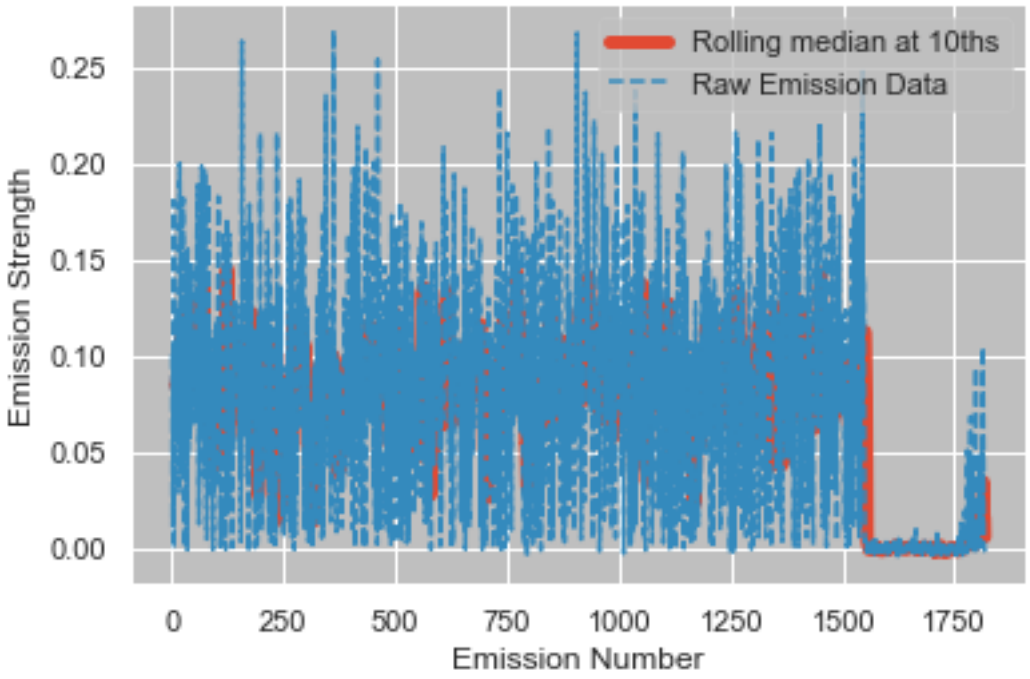
[ ]:
```
pulsar['RollingMeanEmissions10ths'] = pulsar["Brightness"].rolling(10).mean()

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	0.101127	0.001893	1	NaN	
1	2	0.012166	0.001814	0	NaN	
2	3	0.021918	0.001835	0	NaN	
3	4	0.181179	0.002183	1	NaN	
4	5	0.000240	0.001725	0	0.063326	
5	6	0.085866	0.001723	1	0.060274	
6	7	0.067280	0.001778	0	0.071297	
7	8	0.092884	0.002438	1	0.085490	
8	9	0.083350	0.002101	1	0.065924	
9	10	0.087871	0.001941	1	0.083450	
10	11	0.123529	0.002026	1	0.090983	
11	12	0.097413	0.001878	1	0.097009	
12	13	0.100649	0.001820	1	0.098562	
13	14	0.058025	0.001724	0	0.093498	
14	15	0.116164	0.001948	1	0.099156	
15	16	0.029203	0.001918	0	0.080291	
16	17	0.174895	0.002131	1	0.095787	
17	18	0.200468	0.002571	1	0.115751	
18	19	0.123890	0.001805	1	0.128924	
19	20	0.083496	0.001856	1	0.122391	
20	21	0.042757	0.001891	0	0.125101	

```
21            22    0.119953     0.001744       1                  0.114113
22            23    0.096266     0.001911       1                  0.093273
23            24    0.040698     0.001975       0                  0.076634
24            25    0.175852     0.002251       1                  0.095105

    RollingMeanEmissions10ths  RollingMedianEmissions5ths  \
0                         NaN                         NaN
1                         NaN                         NaN
2                         NaN                         NaN
3                         NaN                         NaN
4                         NaN                    0.021918
5                         NaN                    0.021918
6                         NaN                    0.067280
7                         NaN                    0.085866
8                         NaN                    0.083350
9                    0.073388                    0.085866
10                   0.075628                    0.087871
11                   0.084153                    0.092884
12                   0.092026                    0.097413
13                   0.079711                    0.097413
14                   0.091303                    0.100649
15                   0.085637                    0.097413
16                   0.096398                    0.100649
17                   0.107157                    0.116164
18                   0.111211                    0.123890
19                   0.110773                    0.123890
20                   0.102696                    0.123890
21                   0.104950                    0.119953
22                   0.104512                    0.096266
23                   0.102779                    0.083496
24                   0.108748                    0.096266

    RollingMedianEmissions10ths
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN
5                           NaN
6                           NaN
7                           NaN
8                           NaN
9                      0.084608
10                     0.084608
11                     0.086868
12                     0.090377
13                     0.086868
```

```
14                    0.090377
15                    0.090377
16                    0.095148
17                    0.099031
18                    0.108406
19                    0.108406
20                    0.099031
21                    0.108406
22                    0.106215
23                    0.106215
24                    0.108110
```

4.1 Binary Classification

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

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

[ ]:
```
   Brightness  Uncertainty
0    0.101127     0.001893
1    0.012166     0.001814
2    0.021918     0.001835
3    0.181179     0.002183
4    0.000240     0.001725
```

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

[ ]:
```
0    1
1    0
2    0
3    1
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)  =  185
False Positive(FP) =  0
True Negative(TN)  =  175
False Negative(FN) =  4
```

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

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

```
Accuracy of the model is  0.989010989010989
```

4.2 Bidirectional LSTM Model

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

[ ]:
```
[[0.1011271, 0.001893053],
 [0.01216605, 0.00181368],
 [0.02191846, 0.001835275],
 [0.1811794, 0.002183303],
 [0.0002404589, 0.001724854],
 [0.08586562, 0.001723405],
 [0.06727986, 0.001777844],
 [0.09288353, 0.002437727],
 [0.08335005, 0.002100959],
 [0.08787134, 0.001940818]]
```

[ ]:
```
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
```

[ ]:
```
(1375, 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
51/51 [==============================] - 6s 41ms/step - loss: 6.8462 - accuracy:
0.0000e+00
Epoch 2/50
```

```
51/51 [==============================] - 2s 46ms/step - loss: 0.6412 - accuracy:
0.0000e+00
Epoch 3/50
51/51 [==============================] - 2s 47ms/step - loss: 0.6184 - accuracy:
0.0000e+00
Epoch 4/50
51/51 [==============================] - 2s 44ms/step - loss: 0.5938 - accuracy:
0.0000e+00
Epoch 5/50
51/51 [==============================] - 2s 45ms/step - loss: 3428246.0000 -
accuracy: 0.0000e+00
Epoch 6/50
51/51 [==============================] - 2s 45ms/step - loss: 0.5659 - accuracy:
0.0000e+00
Epoch 7/50
51/51 [==============================] - 2s 46ms/step - loss: 0.5493 - accuracy:
0.0000e+00
Epoch 8/50
51/51 [==============================] - 3s 51ms/step - loss: 0.7532 - accuracy:
0.0000e+00
Epoch 9/50
51/51 [==============================] - 2s 47ms/step - loss: 0.5154 - accuracy:
0.0000e+00
Epoch 10/50
51/51 [==============================] - 2s 40ms/step - loss: 0.6373 - accuracy:
0.0000e+00
Epoch 11/50
51/51 [==============================] - 2s 44ms/step - loss: 0.5075 - accuracy:
0.0000e+00
Epoch 12/50
51/51 [==============================] - 2s 47ms/step - loss: 0.4726 - accuracy:
0.0000e+00
Epoch 13/50
51/51 [==============================] - 2s 49ms/step - loss: 0.4601 - accuracy:
0.0000e+00
Epoch 14/50
51/51 [==============================] - 2s 43ms/step - loss: 0.4479 - accuracy:
0.0000e+00
Epoch 15/50
51/51 [==============================] - 2s 45ms/step - loss: 0.4327 - accuracy:
0.0000e+00
Epoch 16/50
51/51 [==============================] - 2s 48ms/step - loss: 0.4248 - accuracy:
0.0000e+00
Epoch 17/50
51/51 [==============================] - 2s 45ms/step - loss: 0.4142 - accuracy:
0.0000e+00
Epoch 18/50
```

```
51/51 [==============================] - 2s 46ms/step - loss: 0.4040 - accuracy:
0.0000e+00
Epoch 19/50
51/51 [==============================] - 2s 47ms/step - loss: 0.3942 - accuracy:
0.0000e+00
Epoch 20/50
51/51 [==============================] - 2s 46ms/step - loss: 0.3848 - accuracy:
0.0000e+00
Epoch 21/50
51/51 [==============================] - 2s 46ms/step - loss: 0.3758 - accuracy:
0.0000e+00
Epoch 22/50
51/51 [==============================] - 2s 45ms/step - loss: 0.3670 - accuracy:
0.0000e+00
Epoch 23/50
51/51 [==============================] - 2s 45ms/step - loss: 0.3586 - accuracy:
0.0000e+00
Epoch 24/50
51/51 [==============================] - 2s 45ms/step - loss: 0.3505 - accuracy:
0.0000e+00
Epoch 25/50
51/51 [==============================] - 3s 49ms/step - loss: 0.3427 - accuracy:
0.0000e+00
Epoch 26/50
51/51 [==============================] - 2s 40ms/step - loss: 0.3352 - accuracy:
0.0000e+00
Epoch 27/50
51/51 [==============================] - 2s 47ms/step - loss: 0.3277 - accuracy:
0.0000e+00
Epoch 28/50
51/51 [==============================] - 2s 46ms/step - loss: 0.3205 - accuracy:
0.0000e+00
Epoch 29/50
51/51 [==============================] - 2s 47ms/step - loss: 0.3134 - accuracy:
0.0000e+00
Epoch 30/50
51/51 [==============================] - 2s 46ms/step - loss: 0.3065 - accuracy:
0.0000e+00
Epoch 31/50
51/51 [==============================] - 2s 45ms/step - loss: 0.2994 - accuracy:
0.0000e+00
Epoch 32/50
51/51 [==============================] - 2s 45ms/step - loss: 0.2920 - accuracy:
0.0000e+00
Epoch 33/50
51/51 [==============================] - 2s 41ms/step - loss: 34.0462 -
accuracy: 0.0000e+00
Epoch 34/50
```

```
51/51 [==============================] - 2s 46ms/step - loss: 0.2980 - accuracy:
0.0000e+00
Epoch 35/50
51/51 [==============================] - 2s 47ms/step - loss: 0.2910 - accuracy:
0.0000e+00
Epoch 36/50
51/51 [==============================] - 2s 46ms/step - loss: 0.2827 - accuracy:
0.0000e+00
Epoch 37/50
51/51 [==============================] - 2s 44ms/step - loss: 0.2780 - accuracy:
0.0000e+00
Epoch 38/50
51/51 [==============================] - 2s 47ms/step - loss: 0.2699 - accuracy:
0.0000e+00
Epoch 39/50
51/51 [==============================] - 2s 45ms/step - loss: 0.2604 - accuracy:
0.0000e+00
Epoch 40/50
51/51 [==============================] - 2s 47ms/step - loss: 0.4537 - accuracy:
0.0000e+00
Epoch 41/50
51/51 [==============================] - 2s 48ms/step - loss: 0.2655 - accuracy:
0.0000e+00
Epoch 42/50
51/51 [==============================] - 2s 48ms/step - loss: 0.2608 - accuracy:
0.0000e+00
Epoch 43/50
51/51 [==============================] - 2s 42ms/step - loss: 0.2566 - accuracy:
0.0000e+00
Epoch 44/50
51/51 [==============================] - 2s 46ms/step - loss: 0.2525 - accuracy:
0.0000e+00
Epoch 45/50
51/51 [==============================] - 2s 45ms/step - loss: 0.2484 - accuracy:
0.0000e+00
Epoch 46/50
51/51 [==============================] - 2s 44ms/step - loss: 0.2447 - accuracy:
0.0000e+00
Epoch 47/50
51/51 [==============================] - 2s 38ms/step - loss: 0.2412 - accuracy:
0.0000e+00
Epoch 48/50
51/51 [==============================] - 2s 32ms/step - loss: 0.2379 - accuracy:
0.0000e+00
Epoch 49/50
51/51 [==============================] - 2s 31ms/step - loss: 0.2345 - accuracy:
0.0000e+00
Epoch 50/50
```

```
51/51 [==============================] - 2s 33ms/step - loss: 0.2315 - 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([[0.8742558 ],
       [0.29719484],
       [0.87434995],
       [0.39804944],
       [0.87422234],
       [0.6589098 ],
       [0.864233  ],
       [0.25351036],
       [0.16653153],
       [0.8743218 ]], dtype=float32)
```

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

```
11/11 [==============================] - 0s 8ms/step - loss: 0.2446 - accuracy:
0.0000e+00
```

[ ]:
```
[0.2445821613073349, 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 corresponding 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:  [1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,
0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1,
0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0,
1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1,
1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
```

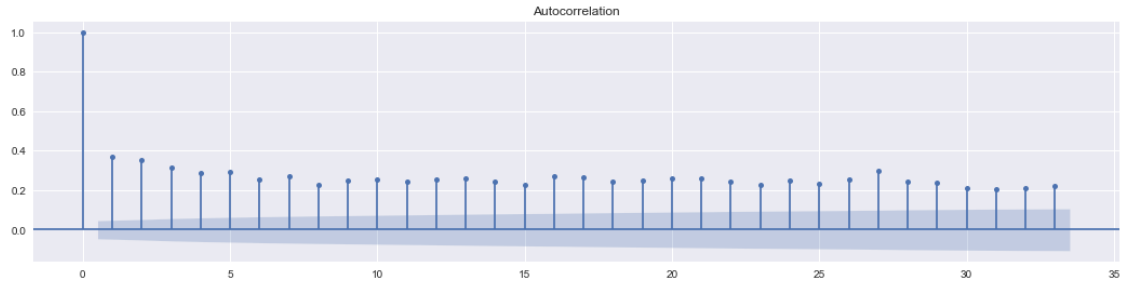
```
1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,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

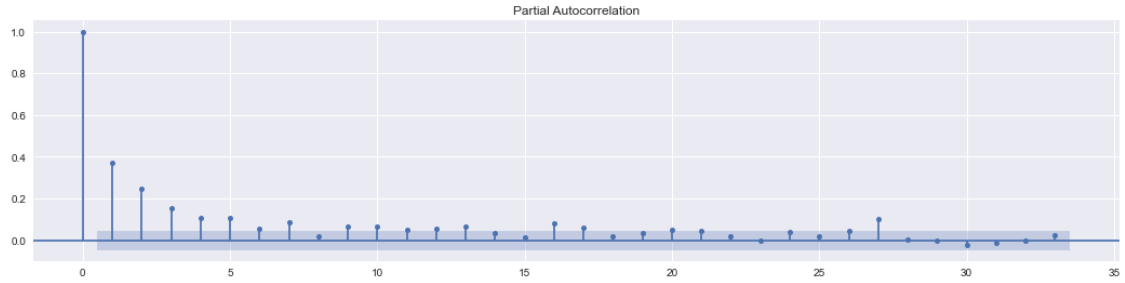
7 Below we begin autocorrelation and autocovariance analysis

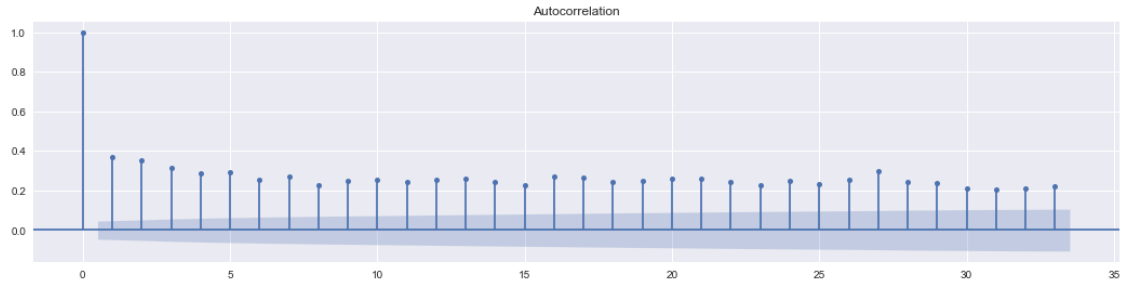
To get started with this I am playing around with guide from: https://towardsdatascience.com/a-step-by-step-guide-to-calculating-autocorrelation-and-partial-autocorrelation-8c4342b784e8

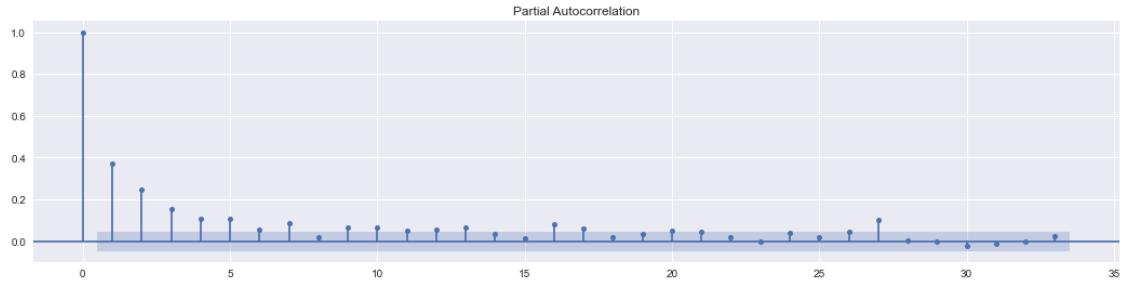
[ ]:
```
plt.style.use("seaborn")
plt.rcParams["figure.figsize"] = (18, 9)

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.        , 0.37138454, 0.34994166, 0.31194031, 0.28665069,
       0.29048719, 0.25431929, 0.27167022, 0.22662943, 0.24809334,
```

```
       0.25146666])
```

[ ]:
```
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   B_lag_6  \
0     0.101127       NaN       NaN       NaN       NaN       NaN       NaN
1     0.012166  0.101127       NaN       NaN       NaN       NaN       NaN
2     0.021918  0.012166  0.101127       NaN       NaN       NaN       NaN
3     0.181179  0.021918  0.012166  0.101127       NaN       NaN       NaN
4     0.000240  0.181179  0.021918  0.012166  0.101127       NaN       NaN
…          …         …         …         …         …         …         …
1814  0.105178  0.008539  0.053246  0.024587  0.004085  0.000947  0.044895
1815  0.064272  0.105178  0.008539  0.053246  0.024587  0.004085  0.000947
1816  0.000171  0.064272  0.105178  0.008539  0.053246  0.024587  0.004085
1817 -0.000924  0.000171  0.064272  0.105178  0.008539  0.053246  0.024587
1818  0.000001 -0.000924  0.000171  0.064272  0.105178  0.008539  0.053246

       B_lag_7   B_lag_8   B_lag_9  B_lag_10
0          NaN       NaN       NaN       NaN
1          NaN       NaN       NaN       NaN
2          NaN       NaN       NaN       NaN
3          NaN       NaN       NaN       NaN
4          NaN       NaN       NaN       NaN
…          …         …         …         …
1814  0.007906  0.048652  0.013009  0.006294
1815  0.044895  0.007906  0.048652  0.013009
1816  0.000947  0.044895  0.007906  0.048652
1817  0.004085  0.000947  0.044895  0.007906
1818  0.024587  0.004085  0.000947  0.044895

[1819 rows x 11 columns]
```

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

[ ]:
```
array([1.        , 0.37158343, 0.35041747, 0.31258703, 0.28752434,
       0.29153195, 0.25533259, 0.27276504, 0.22759855, 0.2492633 ,
       0.25277541])
```

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	0.101127	0.001893	1	NaN	
5	6	0.085866	0.001723	1	0.060274	
10	11	0.123529	0.002026	1	0.090983	
15	16	0.029203	0.001918	0	0.080291	
20	21	0.042757	0.001891	0	0.125101	
…	…	…	…	…	…	
1795	1796	0.004570	0.001779	0	0.010804	
1800	1801	0.002429	0.001749	0	0.042189	
1805	1806	0.013009	0.001764	0	0.005799	
1810	1811	0.004085	0.001713	0	0.021297	
1815	1816	0.064272	0.001995	0	0.051165	

	RollingMeanEmissions10ths	RollingMedianEmissions5ths	\
0	NaN	NaN	
5	NaN	0.021918	
10	0.075628	0.087871	
15	0.085637	0.097413	
20	0.102696	0.123890	
…	…	…	
1795	0.015280	0.002436	
1800	0.026497	0.056999	
1805	0.023994	0.006294	
1810	0.013548	0.007906	
1815	0.036231	0.053246	

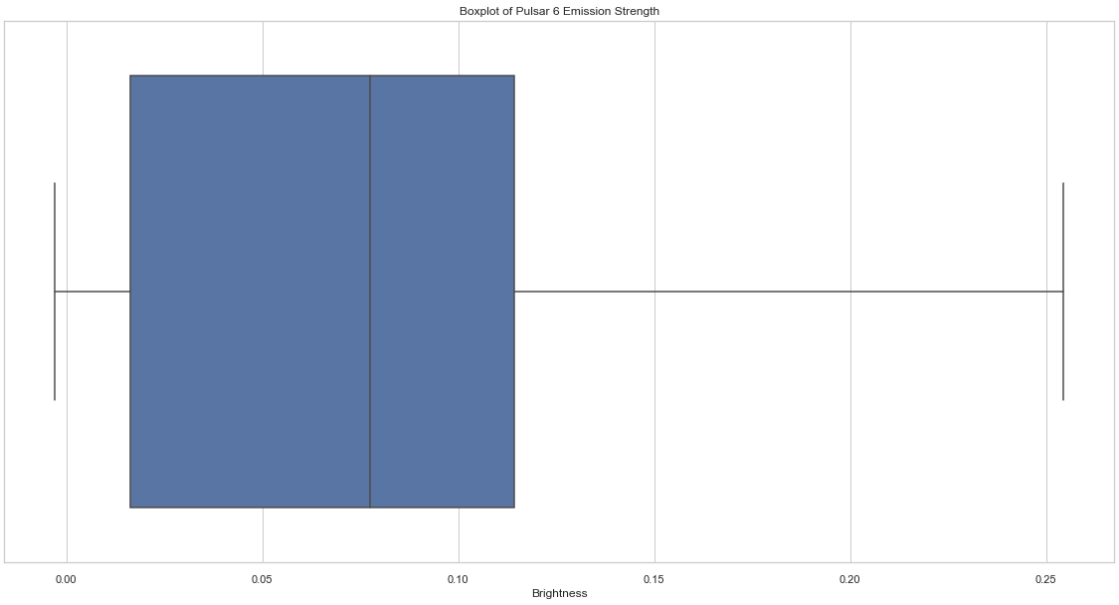
	RollingMedianEmissions10ths
0	NaN
5	NaN
10	0.084608
15	0.090377
20	0.099031
…	…
1795	0.001989
1800	0.003503
1805	0.007042
1810	0.007042
1815	0.034741

```
[364 rows x 8 columns]
```

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

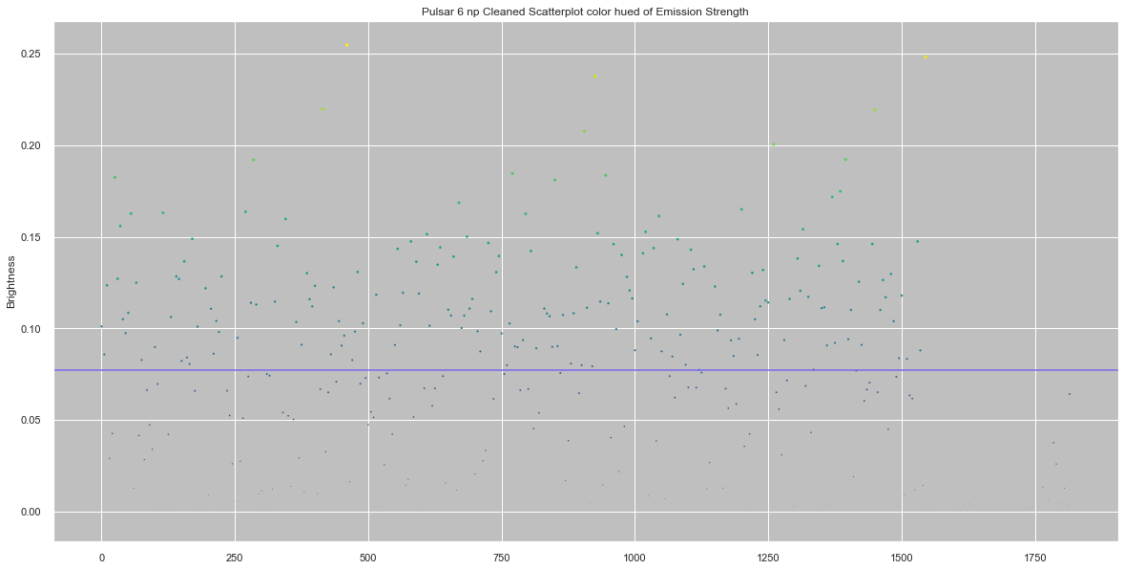
[ ]:
```
0.07756883
```

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

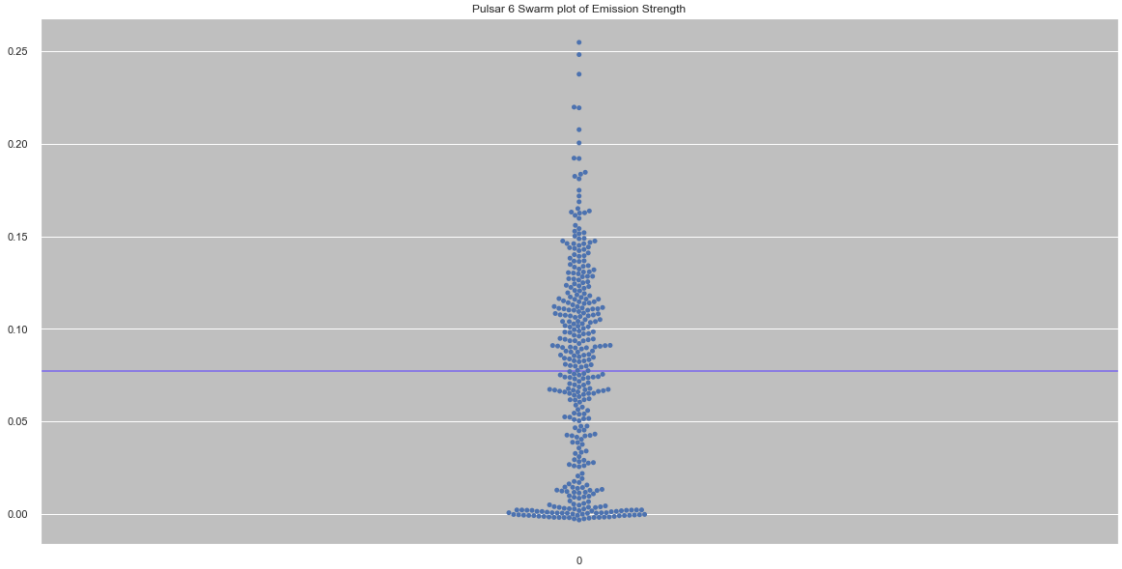


[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = held5ths.Brightness.values
ax = sns.scatterplot(data=held5ths["Brightness"], s= strength*50, c=strength,␣
↪cmap="viridis", marker="o").set_title('Pulsar 6 np Cleaned Scatterplot color␣
↪hued of Emission Strength')
ax = plt.axhline( y=0.07756883, 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
```



[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = held5ths.Brightness.values
ax = plt.axhline( y=0.07756883, ls='-',c='mediumslateblue')
ax = sns.swarmplot(data=held5ths["Brightness"], c="blue").set_title('Pulsar 6␣
↪Swarm plot of Emission Strength')
```



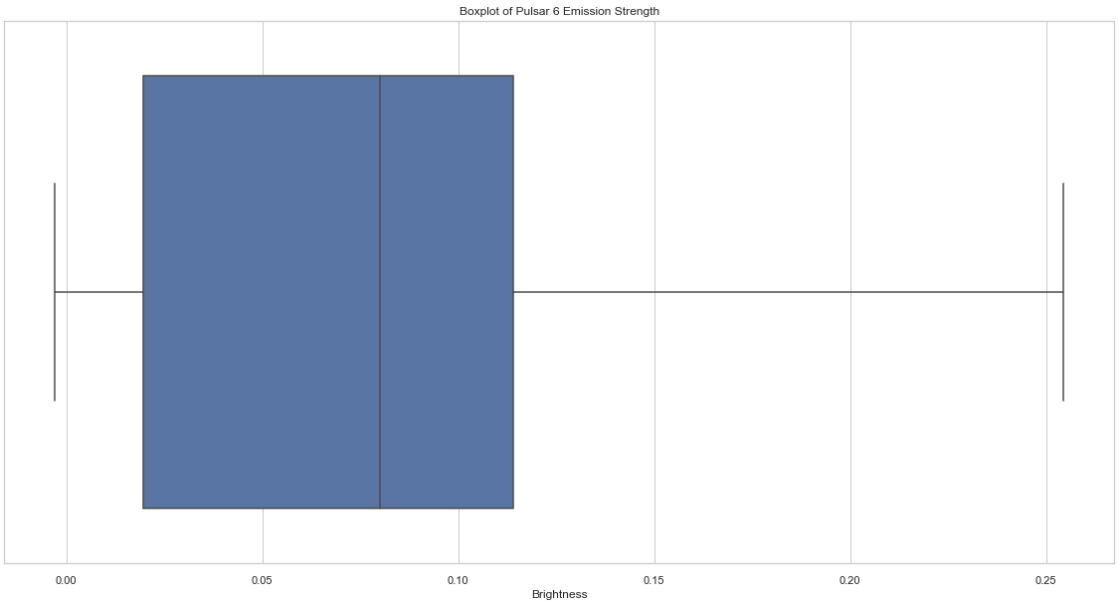
[ ]:
```
print(len(held5ths[(held5ths.Brightness > 0.07756883)]))
print(len(held5ths[(held5ths.Brightness < 0.07756883)]))
```

```
182
182
```

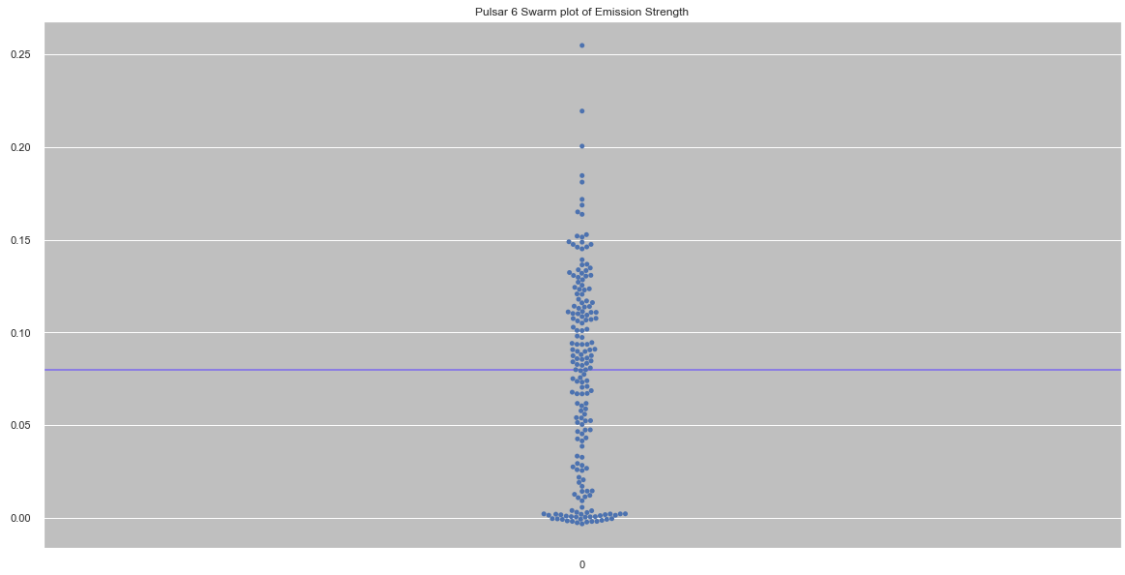
[ ]:
```
held10ths = pulsar[pulsar.index % 10 == 0]
medianheld10ths = held10ths["Brightness"].median()
medianheld10ths
```

[ ]:
```
0.079977185
```

[ ]:
```
plt.figure(figsize=(20,10))
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=held10ths["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 = plt.axhline( y=0.079977185, ls='-',c='mediumslateblue')
ax = sns.swarmplot(data=held10ths["Brightness"], c="blue").set_title('Pulsar 6␣
↪Swarm plot of Emission Strength')
```



[ ]:
```
print(len(held10ths[(held10ths.Brightness > 0.079977185)]))
print(len(held10ths[(held10ths.Brightness < 0.079977185)]))
```

```
91
91
```

Randomness testing

[ ]:
```
np.savetxt(r'every5thbinarypulsar4.txt', held5ths.Binary, fmt='%d',␣
 ↪delimiter='')
np.savetxt(r'allpulsar4.txt', pulsar.Binary, fmt='%d', delimiter='')
np.savetxt(r'every10thbinarypulsar4.txt', held10ths.Binary, fmt='%d',␣
 ↪delimiter='')
```

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

[ ]:
```
0       1
1       0
2       0
3       1
4       0
       ..
1814    1
1815    0
1816    0
1817    0
1818    0
Name: Binary, Length: 1819, dtype: int32
```