pulsar1

October 20, 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/J0437-4715.pulses", sep = ' ', header = None, names␣
↪= colnames)
```

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

[ ]:
```
(27000, 3)
```

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

[ ]:

	Pulse Number	Brightness	Uncertainty
0	1	0.598393	0.056431
1	2	0.590859	0.055182
2	3	0.449643	0.063632
3	4	0.682860	0.056269
4	5	0.490026	0.046830
5	6	0.586071	0.052649
6	7	0.150353	0.056483
7	8	0.384684	0.052567
8	9	0.429094	0.055569
9	10	0.995865	0.075811
10	11	0.670907	0.049539
11	12	0.465406	0.047461
12	13	0.242442	0.050653
13	14	0.500057	0.050163
14	15	0.658159	0.050743
15	16	0.404870	0.056679
16	17	0.595339	0.065296
17	18	0.230061	0.051813
18	19	0.423335	0.049558

```
19             20    0.208840    0.049900
20             21    0.297223    0.048826
21             22    0.749683    0.071350
22             23    0.387574    0.054314
23             24    0.466527    0.045075
24             25    1.333974    0.092806
```

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

[ ]:
```
       Pulse Number    Brightness   Uncertainty
count   27000.00000  27000.000000  27000.000000
mean    13500.50000      0.536400      0.062556
std      7794.37297      0.413764      0.056313
min         1.00000     -5.114133      0.015426
25%      6750.75000      0.296443      0.052381
50%     13500.50000      0.423816      0.056856
75%     20250.25000      0.643723      0.063111
max     27000.00000     18.722410      3.049559
```

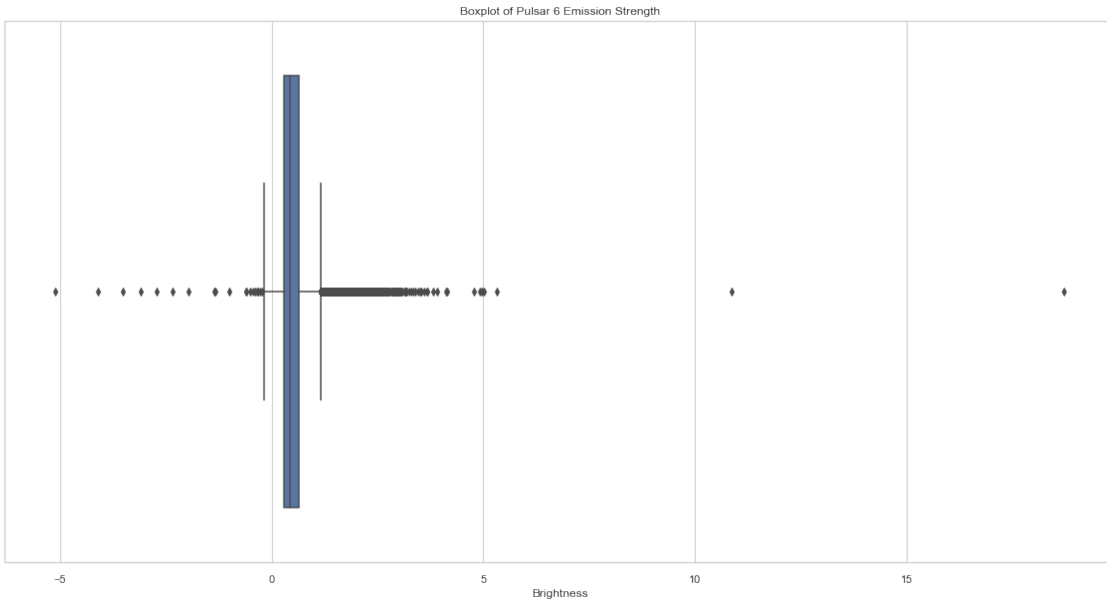
[ ]:
```
nullBoolBrightness = pd.isnull(pulsar["Brightness"])

pulsar[nullBoolBrightness]
```

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

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

[ ]:
```
count    27000.000000
mean         0.536400
std          0.413764
min         -5.114133
25%          0.296443
50%          0.423816
75%          0.643723
max         18.722410
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:  0.42381595
```

[ ]:
```
pulsar
```

[ ]:

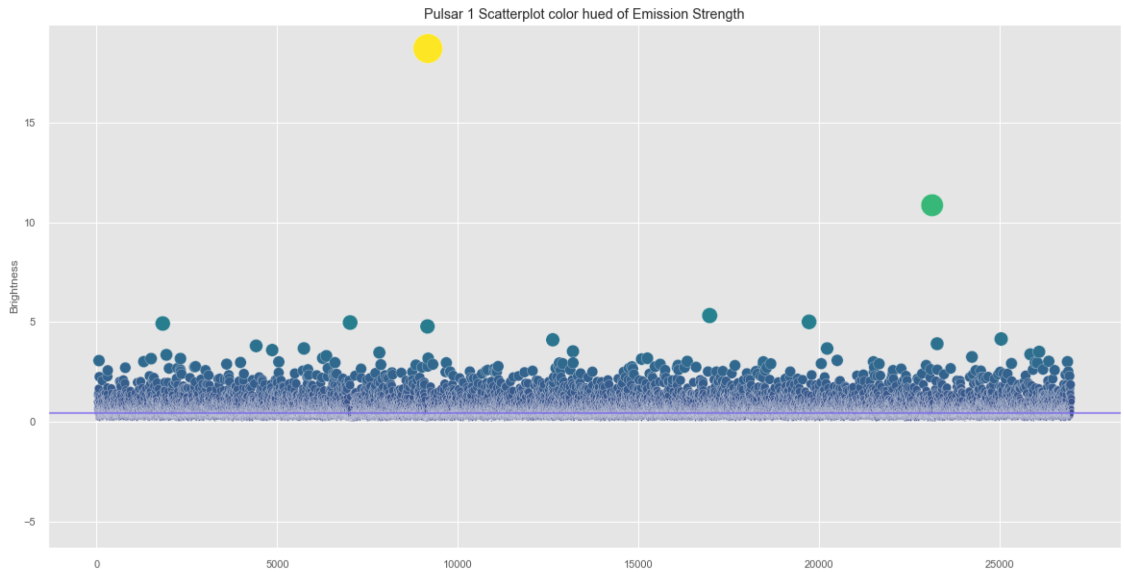
	Pulse Number	Brightness	Uncertainty	Binary
0	1	0.598393	0.056431	1
1	2	0.590859	0.055182	1
2	3	0.449643	0.063632	1
3	4	0.682860	0.056269	1
4	5	0.490026	0.046830	1
…	…	…	…	…
26995	26996	0.539079	0.063854	1
26996	26997	0.324070	0.054332	0
26997	26998	0.291341	0.058106	0
26998	26999	0.346267	0.058064	0
26999	27000	0.513315	0.064349	1

[27000 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 1 Scatterplot color hued of␣
↪Emission Strength')
ax= plt.axhline( y=0.42381595, ls='-',c='mediumslateblue')
```

```
c:\Users\oxlay\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 > 0.42381595)]))
print(len(pulsar[(pulsar.Brightness < 0.42381595)]))
```

```
13500
13500
```

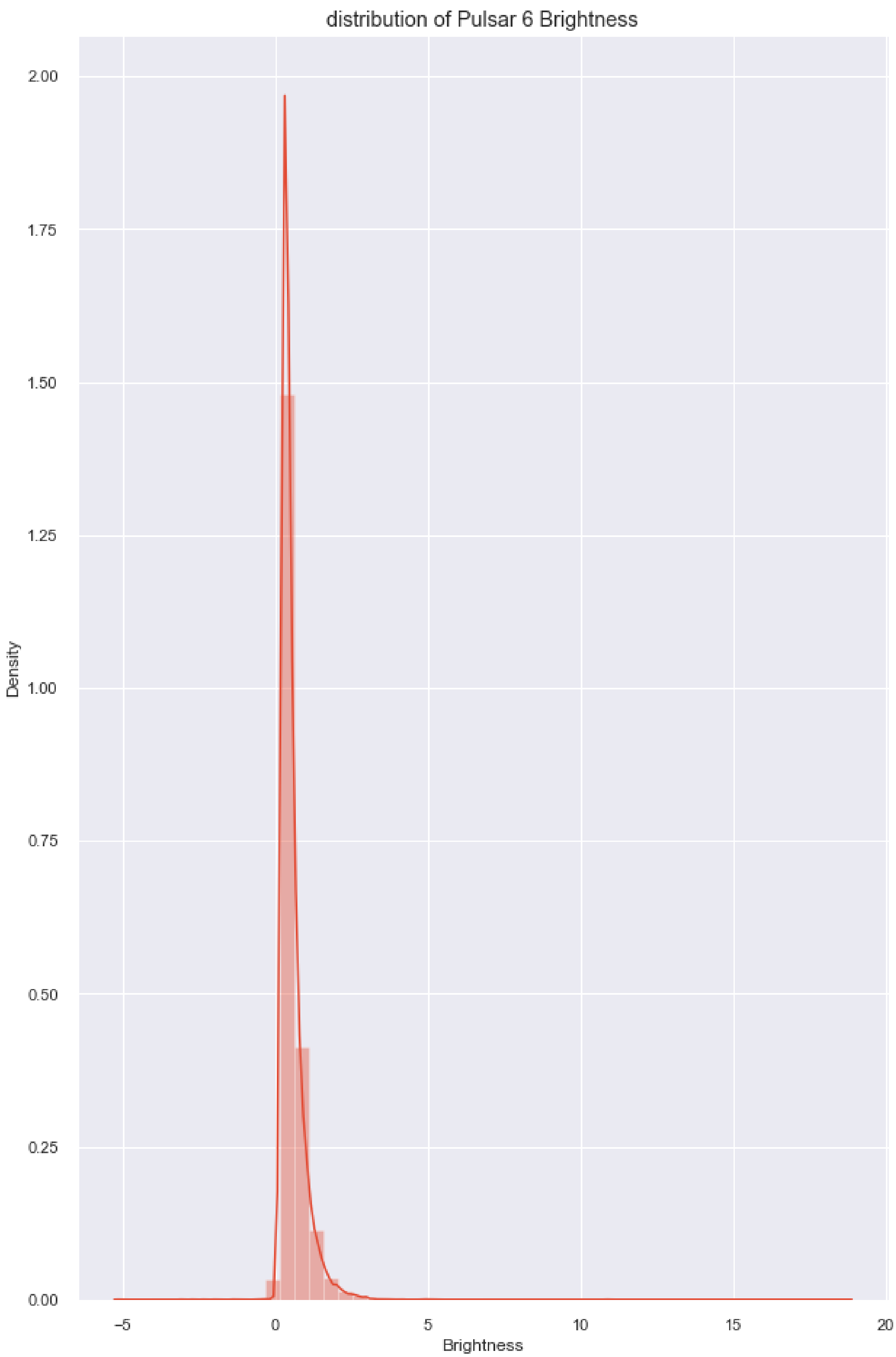
[ ]:
```
#plt.figure(figsize=(20,10))
#sns.set_style("darkgrid", {"axes.facecolor": ".75"})
#strength = pulsar.Brightness.values
#ax = plt.axhline( y=0.65829515, ls='-',c='mediumslateblue')
#ax = sns.swarmplot(data=pulsar["Brightness"], c="blue").set_title('Pulsar 6␣
↪Swarm plot of Emission Strength')
```

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

```
c:\Users\oxlay\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)
```

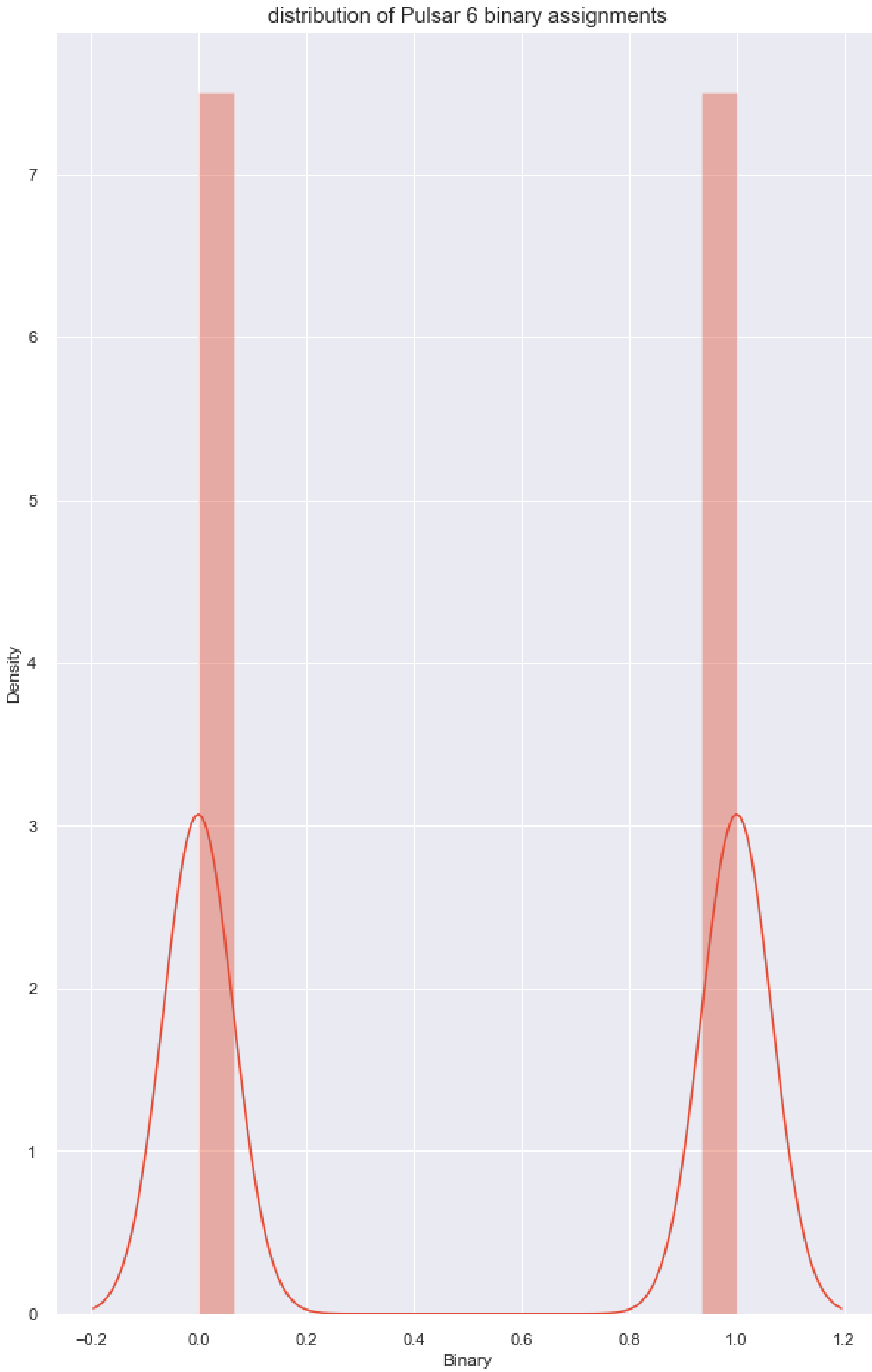
[ ]:
```
Text(0.5, 1.0, 'distribution of Pulsar 6 Brightness')
```


distribution of Pulsar 6 Brightness
Density
Brightness
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−5
0
5
10
15
20

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

```
c:\Users\oxlay\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')
```

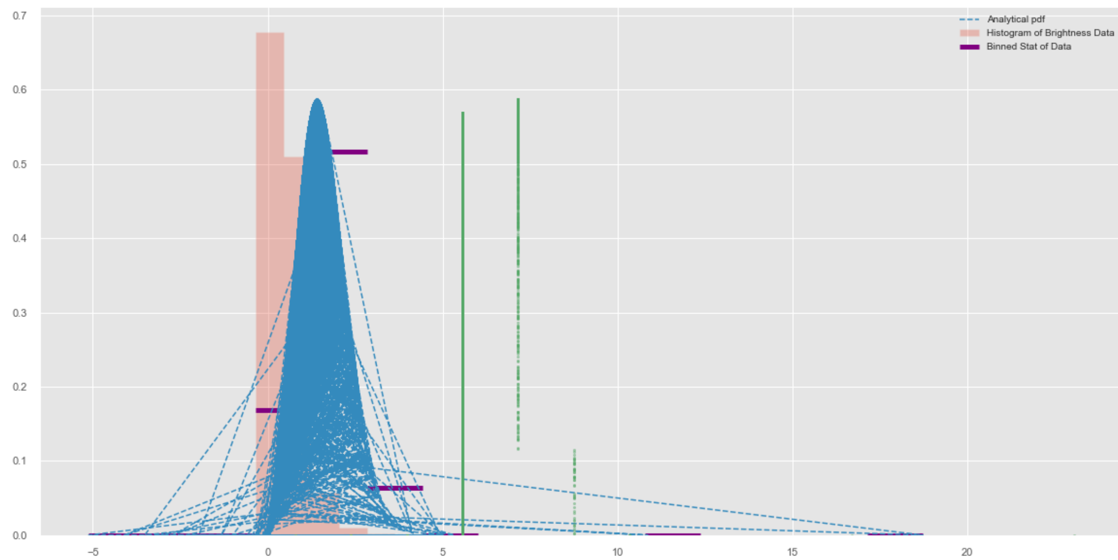

distribution of Pulsar 6 binary assignments
Density
Binary
0
1
2
3
4
5
6
7
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

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

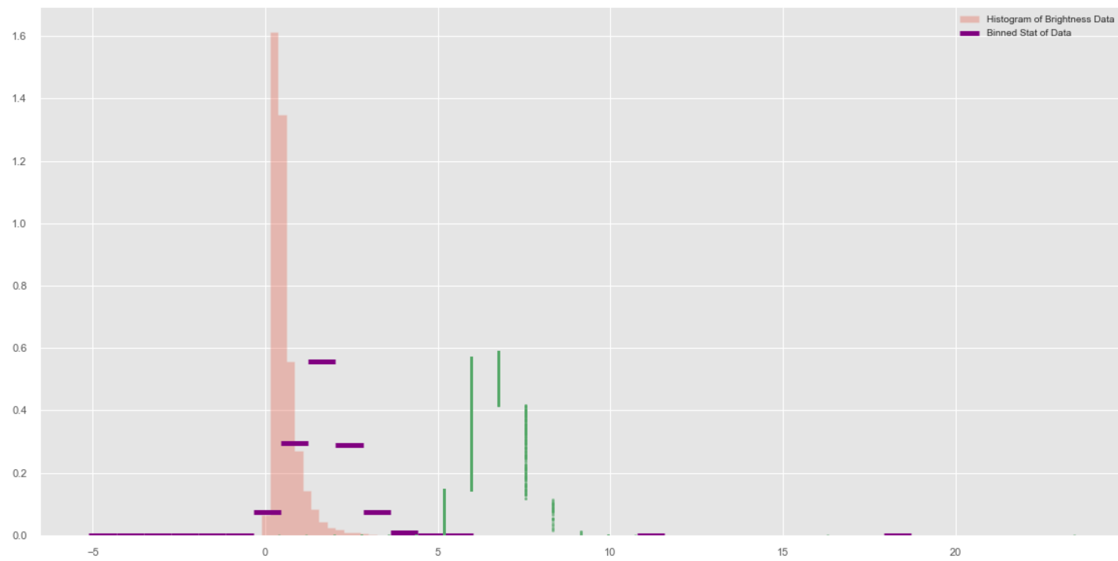
[ ]:
```
data = pulsar["Brightness"]
data
```

[ ]:
```
0        0.598393
1        0.590859
2        0.449643
3        0.682860
4        0.490026
           …
26995    0.539079
26996    0.324070
26997    0.291341
26998    0.346267
26999    0.513315
Name: Brightness, Length: 27000, dtype: float64
```

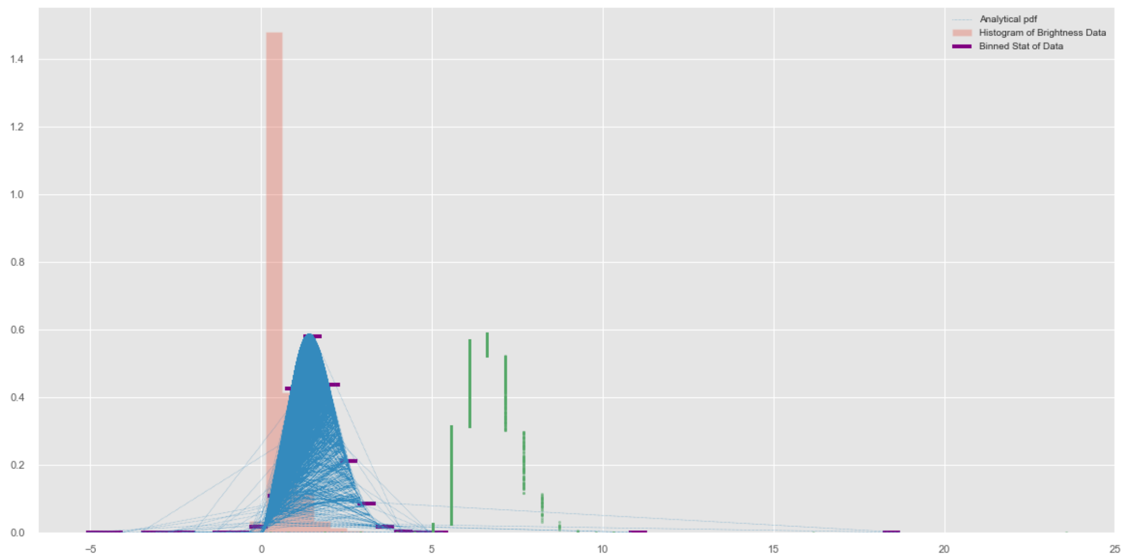
[ ]:
```
dataPDF = stats.maxwell.pdf(data)
bin_means, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='mean', bins=15)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

plt.figure(figsize=(20,10))
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()
```



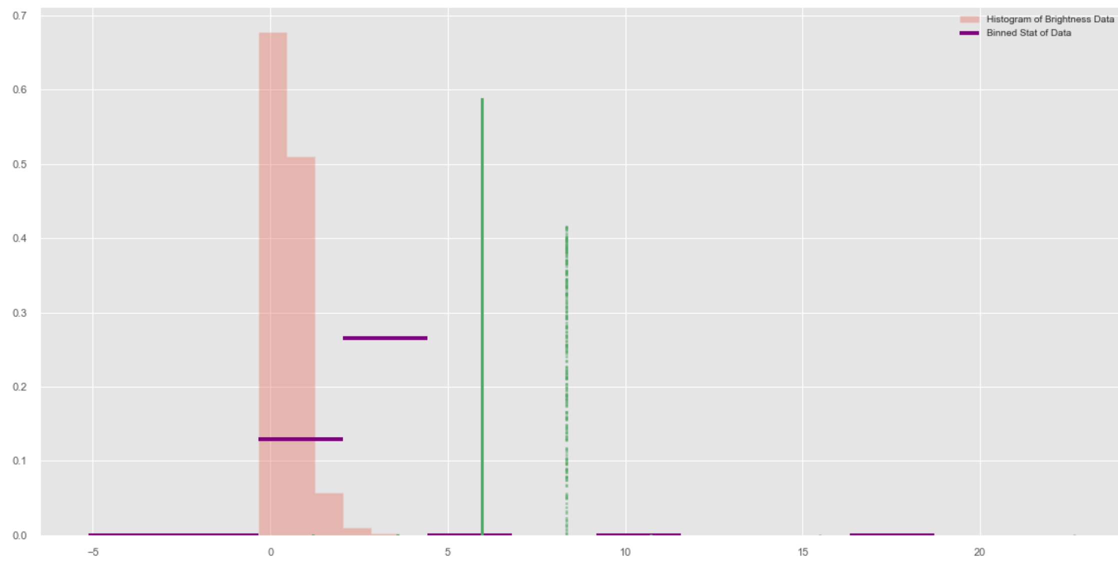
[ ]:
```
dataPDF = stats.maxwell.pdf(data)
bin_means, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='mean', bins=30)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

plt.figure(figsize=(20,10))
plt.hist(data, bins=100, density=True, histtype='stepfilled', alpha=0.3,␣
 ↪label='Histogram of Brightness Data')
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()
```



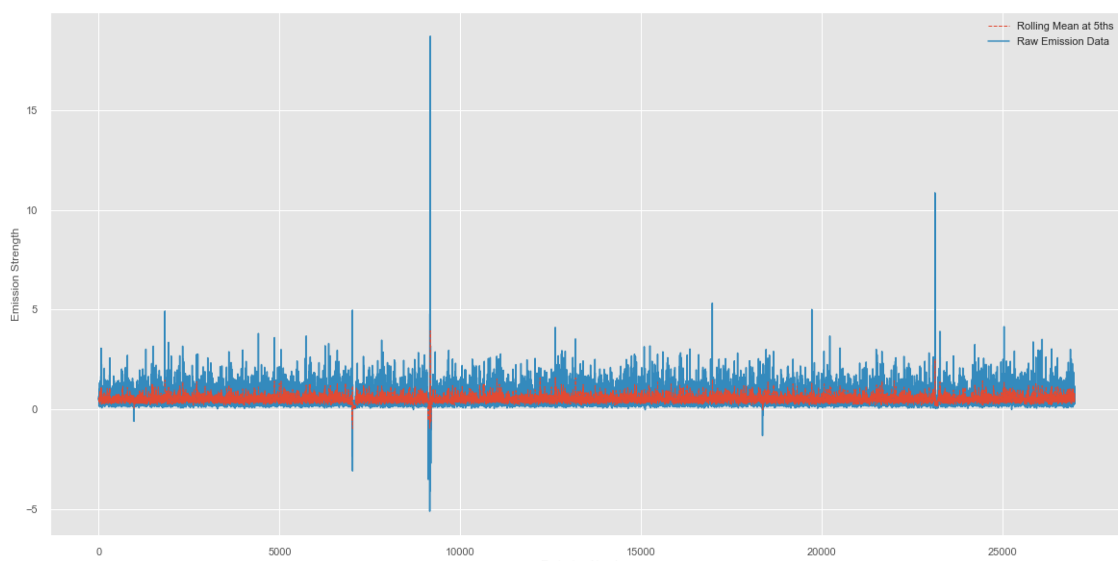
[ ]:
```
bin_median, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='median', bins=45)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

plt.figure(figsize=(20,10))
plt.hist(data, bins=50, 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()
```



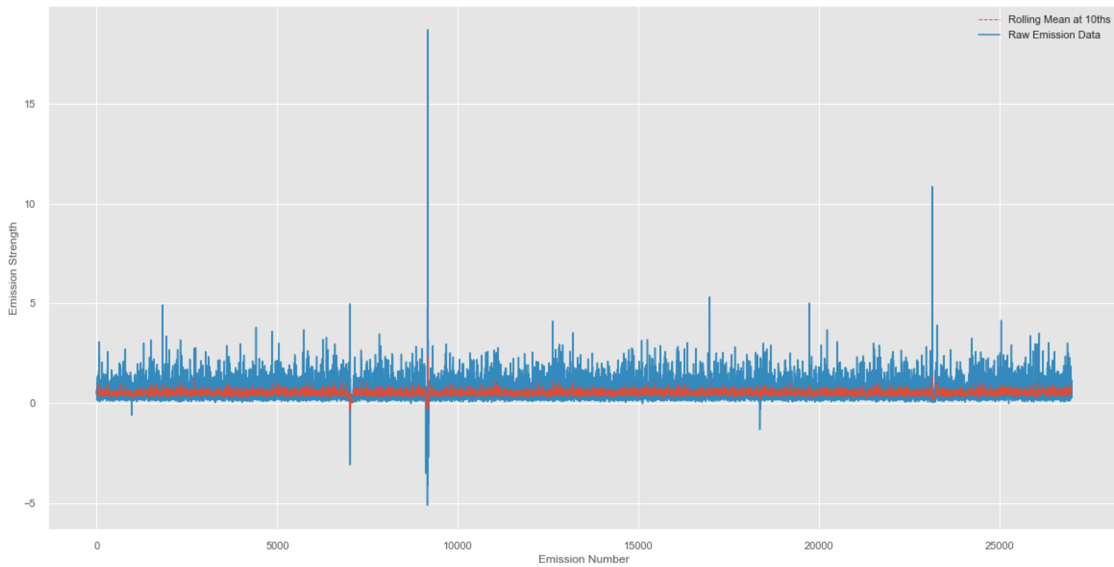

```
[ ]: bin_median, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
             statistic='median', bins=10)
     bin_width = (bin_edges[1] - bin_edges[0])
     bin_centers = bin_edges[1:] - bin_width/2

     plt.figure(figsize=(20,10))
     plt.hist(data, bins=30, density=True, histtype='stepfilled', alpha=0.3,␣
      ↪label='Histogram of Brightness Data')
     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()
```



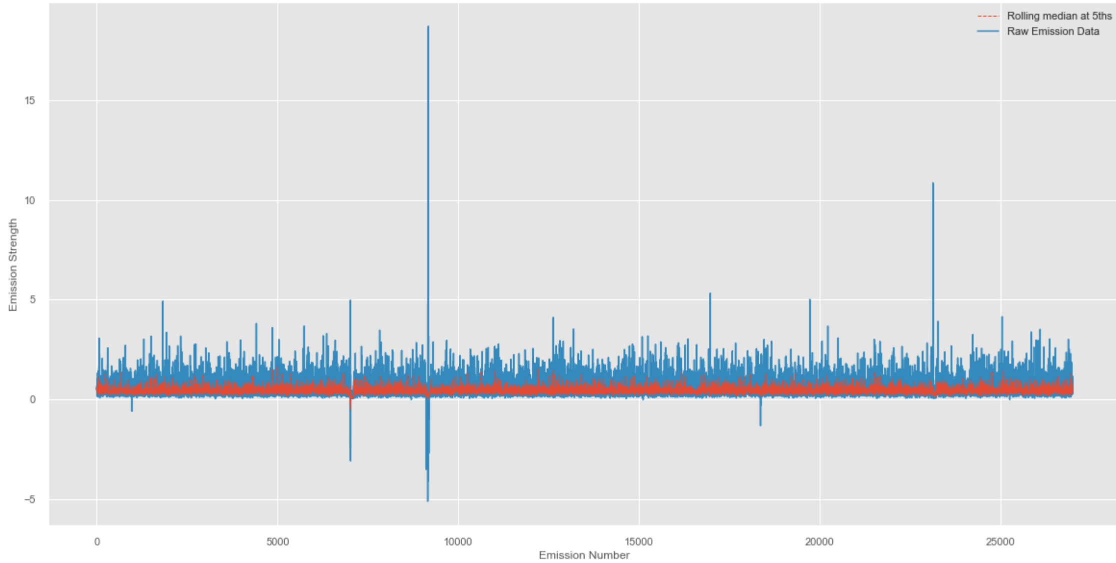
[ ]:
```
pulsar['RollingMeanEmissions5ths'] = pulsar["Brightness"].rolling(5).mean()

plt.figure(figsize=(20,10))
plt.plot(pulsar['RollingMeanEmissions5ths'], label="Rolling Mean at 5ths",␣
 ↪lw=1, linestyle='--', zorder=2)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", zorder=1)
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```



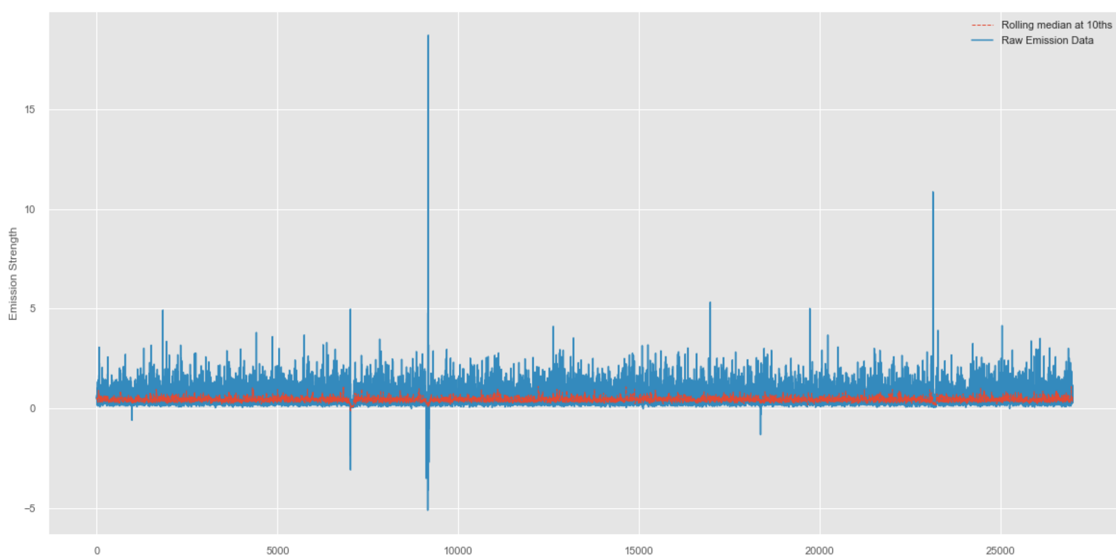
[ ]:
```python
pulsar['RollingMeanEmissions10ths'] = pulsar["Brightness"].rolling(10).mean()

plt.figure(figsize=(20,10))
plt.plot(pulsar['RollingMeanEmissions10ths'], label="Rolling Mean at 10ths",␣
↪lw=1, linestyle='--', zorder=2)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", zorder=1)
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```



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

plt.figure(figsize=(20,10))
plt.plot(pulsar['RollingMedianEmissions5ths'], label="Rolling median at 5ths",␣
↪lw=1, linestyle='--', zorder=2)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", zorder=1)
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```



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

plt.figure(figsize=(20,10))
plt.plot(pulsar['RollingMedianEmissions10ths'], label="Rolling median at␣
 ↪10ths", lw=1, linestyle='--', zorder=2)
plt.plot(pulsar['Brightness'], label= "Raw Emission Data", zorder=1)
plt.legend()
plt.ylabel('Emission Strength')
plt.xlabel('Emission Number')
plt.show()
```



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

```
[ ]:    Pulse Number  Brightness  Uncertainty  Binary  RollingMeanEmissions5ths  \
    0              1    0.598393     0.056431       1                      NaN
    1              2    0.590859     0.055182       1                      NaN
    2              3    0.449643     0.063632       1                      NaN
    3              4    0.682860     0.056269       1                      NaN
    4              5    0.490026     0.046830       1                 0.562356
    5              6    0.586071     0.052649       1                 0.559892
    6              7    0.150353     0.056483       0                 0.471791
    7              8    0.384684     0.052567       0                 0.458799
    8              9    0.429094     0.055569       1                 0.408046
    9             10    0.995865     0.075811       1                 0.509214
    10            11    0.670907     0.049539       1                 0.526181
    11            12    0.465406     0.047461       1                 0.589191
    12            13    0.242442     0.050653       0                 0.560743
    13            14    0.500057     0.050163       1                 0.574935
    14            15    0.658159     0.050743       1                 0.507394
    15            16    0.404870     0.056679       0                 0.454187
    16            17    0.595339     0.065296       1                 0.480173
    17            18    0.230061     0.051813       0                 0.477697
    18            19    0.423335     0.049558       0                 0.462353
    19            20    0.208840     0.049900       0                 0.372489
    20            21    0.297223     0.048826       0                 0.350959
    21            22    0.749683     0.071350       1                 0.381828
    22            23    0.387574     0.054314       0                 0.413331
    23            24    0.466527     0.045075       1                 0.421969
    24            25    1.333974     0.092806       1                 0.646996

        RollingMeanEmissions10ths  RollingMedianEmissions5ths  \
    0                         NaN                         NaN
    1                         NaN                         NaN
    2                         NaN                         NaN
    3                         NaN                         NaN
    4                         NaN                    0.590859
    5                         NaN                    0.586071
    6                         NaN                    0.490026
    7                         NaN                    0.490026
    8                         NaN                    0.429094
    9                    0.535785                    0.429094
    10                   0.543036                    0.429094
    11                   0.530491                    0.465406
    12                   0.509771                    0.465406
    13                   0.491490                    0.500057
```

```
14                     0.508304                     0.500057
15                     0.490184                     0.465406
16                     0.534682                     0.500057
17                     0.519220                     0.500057
18                     0.518644                     0.423335
19                     0.439941                     0.404870
20                     0.402573                     0.297223
21                     0.431001                     0.297223
22                     0.445514                     0.387574
23                     0.442161                     0.387574
24                     0.509742                     0.466527

    RollingMedianEmissions10ths
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN
5                           NaN
6                           NaN
7                           NaN
8                           NaN
9                      0.538048
10                     0.538048
11                     0.477716
12                     0.477716
13                     0.477716
14                     0.482731
15                     0.447250
16                     0.482731
17                     0.482731
18                     0.482731
19                     0.444370
20                     0.414102
21                     0.414102
22                     0.414102
23                     0.414102
24                     0.414102
```

4.1 Binary Classification

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

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

[ ]:

	Brightness	Uncertainty
0	0.598393	0.056431
1	0.590859	0.055182
2	0.449643	0.063632
3	0.682860	0.056269
4	0.490026	0.046830

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

[ ]:
```
0    1
1    1
2    1
3    1
4    1
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)  =  2714
```

```
False Positive(FP) =  151
True Negative(TN)  =  2535
False Negative(FN) =  0
```

```
[ ]: # accuracy, recall, precision and F1

     acc = (2689 + 2603)/(2689 + 2603 + 108 + 0)
     recall = (2689)/(0+2689)
     precision = (2689)/(2689+108)
     f1 = (2)/((1/recall)+ (1/precision))

     print(acc)
     print(recall)
     print(precision)
     print(f1)
```

```
0.98
1.0
0.9613872005720415
0.9803135253372222
```

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

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

```
Accuracy of the model is  0.972037037037037
```

4.2 Bidirectional LSTM Model

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

```
[ ]: [[0.5983928, 0.0564313],
      [0.5908588, 0.05518188],
      [0.4496432, 0.06363222],
      [0.6828599, 0.05626933],
      [0.4900255, 0.04683002],
      [0.5860711, 0.05264937],
      [0.1503529, 0.05648303],
      [0.3846841, 0.0525667],
      [0.4290943, 0.05556898],
      [0.9958652, 0.0758106]]
```

```
[ ]: from sklearn import preprocessing

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

[ ]:
```python
# 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)
```

[ ]:
```python
# 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))
```

[ ]:
```python
# 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)
```

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

[ ]:
```python
# training the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪epochs=2, verbose=1, batch_size=(int(X_train.shape[0]/50)))
```

```
Epoch 1/2
51/51 [==============================] - 16s 321ms/step - loss: 0.0030 - mse:
0.0030 - val_loss: 0.0020 - val_mse: 0.0020
Epoch 2/2
51/51 [==============================] - 16s 321ms/step - loss: 0.0030 - mse:
0.0030 - val_loss: 0.0020 - val_mse: 0.0020
```

[ ]:
```python
X_test
```

[ ]:
```
array([[[0.99553523, 0.09439068],
        [0.98720336, 0.15946638],
        [0.99635322, 0.08532449],
        …,
        [0.99387739, 0.11048857],
```

```
       [0.99641844, 0.08455935],
       [0.8243341 , 0.5661036 ]],

      [[0.99057614, 0.13696315],
       [0.99544821, 0.09530401],
       [0.9947645 , 0.10219393],
       …,
       [0.99408919, 0.10856646],
       [0.94736357, 0.32015975],
       [0.99629784, 0.08596864]],

      [[0.97238382, 0.23338747],
       [0.98368855, 0.17988008],
       [0.97366938, 0.22796476],
       …,
       [0.99671461, 0.08099377],
       [0.98860091, 0.15055979],
       [0.9980507 , 0.06240831]],

      …,

      [[0.99022849, 0.13945445],
       [0.97307765, 0.23047754],
       [0.96903857, 0.24690938],
       …,
       [0.95175352, 0.30686355],
       [0.99599942, 0.08935975],
       [0.99685317, 0.07927013]],

      [[0.94920777, 0.31464997],
       [0.99201513, 0.12611893],
       [0.98908038, 0.14737708],
       …,
       [0.98823669, 0.15293218],
       [0.99274678, 0.12022408],
       [0.99117422, 0.13256569]],

      [[0.99698529, 0.07759083],
       [0.98928962, 0.14596588],
       [0.98073236, 0.19535616],
       …,
       [0.98150505, 0.19143627],
       [0.98911424, 0.14714964],
       [0.98376538, 0.17945941]]])
```

[ ]:

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

```
y_pred[:10]
```

[ ]:
```
array([[0.98082703],
       [0.980616  ],
       [0.9810388 ],
       [0.9872831 ],
       [0.9871915 ],
       [0.9864086 ],
       [0.97754884],
       [0.98505926],
       [0.97542995],
       [0.9771127 ]], dtype=float32)
```

[ ]:
```
y_test[:10]
```

[ ]:
```
array([0.98323453, 0.98127979, 0.9970731 , 0.99812731, 0.99129736,
       0.98806795, 0.99697155, 0.98026025, 0.99583156, 0.99858726])
```

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

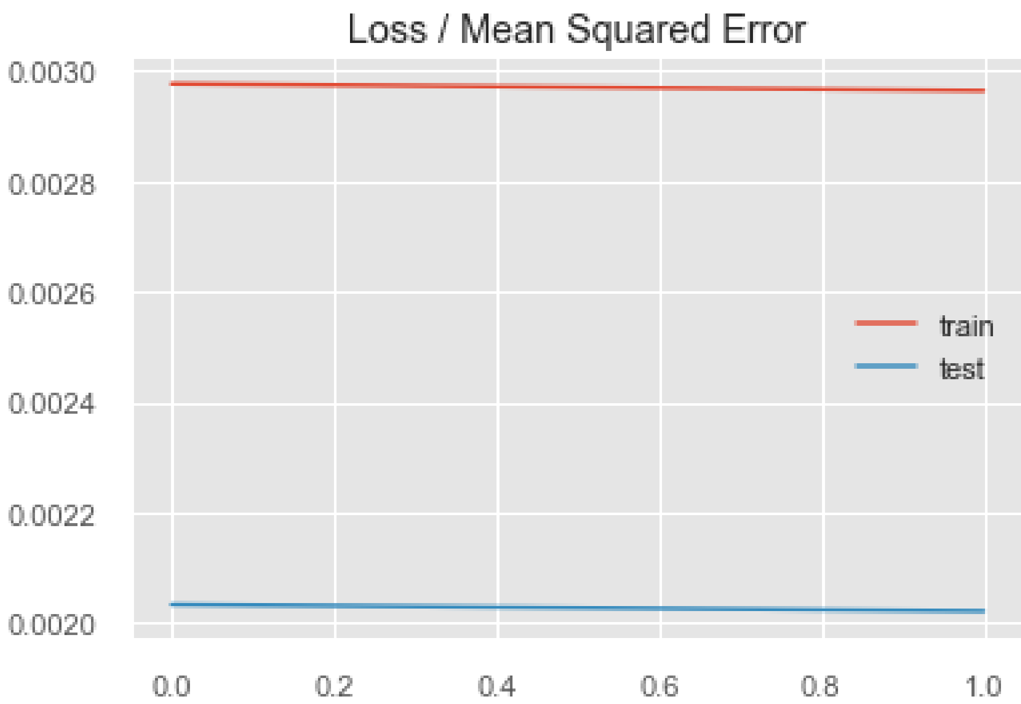
```
169/169 [==============================] - 1s 8ms/step - loss: 0.0020 - mse:
0.0020
```

[ ]:
```
[0.002021806314587593, 0.002021806314587593]
```

[ ]:
```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import math
print('R2 Score: ', r2_score(y_test, y_pred))
print('MAE: ', mean_absolute_error(y_test, y_pred))
print('RSE: ', math.sqrt(mean_absolute_error(y_test, y_pred)))
```

```
R2 Score:  0.12238752597919222
MAE:  0.014404269431746707
RSE:  0.12001778798056023
```

[ ]:
```
plt.title('Loss / Mean Squared Error')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```



[ ]:
```
colnames = ['Pulse Number', 'Brightness', 'Uncertainty']
pulsar2 = pd.read_csv("Data/J0953+0755.pulses", sep = ' ', header = None, names␣
↪= colnames)

newVals = pulsar2[['Brightness', 'Uncertainty']].values.tolist()
```

[ ]:
```
newVals = preprocessing.normalize(newVals)

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))
```

```
[ ]: X[10]
```

```
[ ]: array([[0.99728504, 0.07363802],
            [0.99484056, 0.1014508 ],
            [0.97886433, 0.20451067],
            [0.9950061 , 0.09981416],
            [0.99704107, 0.07687074],
            [0.99034267, 0.13864126],
            [0.99403905, 0.10902464],
            [0.97556486, 0.21971163],
            [0.99321756, 0.1162707 ],
            [0.97262106, 0.2323968 ],
            [0.98677407, 0.16210165],
            [0.99550155, 0.09474528],
            [0.99032304, 0.13878137],
            [0.99536481, 0.09617121],
            [0.99758869, 0.06940314],
            [0.99659603, 0.08244001],
            [0.98475538, 0.17394493],
            [0.98851646, 0.15111326],
            [0.98184257, 0.18969755],
            [0.99369597, 0.1121085 ],
            [0.99021523, 0.13954858],
            [0.99315489, 0.11680483],
            [0.96742421, 0.2531608 ],
            [0.99408912, 0.10856716],
            [0.97794922, 0.20884282],
            [0.90915773, 0.41645195],
            [0.986062  , 0.16637828],
            [0.99868139, 0.05133689],
            [0.99863995, 0.05213691],
            [0.996267  , 0.0863254 ],
            [0.99668407, 0.08136869],
            [0.99083431, 0.13508279],
            [0.99840436, 0.05646887],
            [0.99655906, 0.08288568],
            [0.97912386, 0.20326455],
            [0.997     , 0.07740156],
            [0.99742162, 0.07176434],
            [0.97318251, 0.23003435],
            [0.9953398 , 0.09642971],
            [0.99394052, 0.10991927],
            [0.99856561, 0.05354181],
            [0.98391764, 0.17862274],
```

```
[0.9867255 , 0.16239701],
[0.99728922, 0.07358135],
[0.99827703, 0.05867682],
[0.95702566, 0.29000327],
[0.99401833, 0.10921337],
[0.99849138, 0.05490861],
[0.99795236, 0.0639616 ],
[0.99611093, 0.08810801],
[0.99301447, 0.11799263],
[0.94848309, 0.31682775],
[0.99400291, 0.10935358],
[0.97038077, 0.24158055],
[0.99488992, 0.10096554],
[0.99879261, 0.04912561],
[0.99840165, 0.05651679],
[0.98119704, 0.19300875],
[0.99376891, 0.11146006],
[0.9771171 , 0.21270207],
[0.9907602 , 0.13562532],
[0.99441057, 0.1055823 ],
[0.99673426, 0.08075157],
[0.97014233, 0.2425363 ],
[0.99209587, 0.12548219],
[0.99836742, 0.05711827],
[0.9966292 , 0.082038  ],
[0.99719853, 0.07480033],
[0.99281156, 0.11968791],
[0.97439463, 0.22484461],
[0.99643716, 0.08433847],
[0.99645002, 0.08418646],
[0.97946366, 0.20162076],
[0.99302772, 0.11788105],
[0.99798412, 0.06346421],
[0.98332288, 0.1818684 ],
[0.74692348, 0.66491001],
[0.98033012, 0.19736478],
[0.99193191, 0.1267718 ],
[0.96842309, 0.2493125 ],
[0.98208246, 0.18845168],
[0.99586797, 0.09081287],
[0.99169268, 0.12862983],
[0.97498416, 0.22227433],
[0.99579632, 0.09159526],
[0.99274855, 0.12020947],
[0.9673274 , 0.25353046],
[0.98848965, 0.1512885 ],
[0.9951548 , 0.0983205 ],
```

```
       [0.9929769 , 0.11830837],
       [0.99006059, 0.14064148],
       [0.98568545, 0.16859475],
       [0.99406248, 0.10881079],
       [0.96454268, 0.26392692],
       [0.97201944, 0.23490042],
       [0.99607629, 0.0884987 ],
       [0.9922033 , 0.12462993],
       [0.98727281, 0.15903586],
       [0.95522373, 0.29588448],
       [0.98349506, 0.18093497]])
```

[ ]:
```
y[10]
```

[ ]:
```
0.9972710466812622
```

[ ]:
```
predOut = model.predict(X, verbose=0)
```

[ ]:
```
predOut[10]
```

[ ]:
```
array([0.98242664], dtype=float32)
```

4.3 ML Evaluation.

4.3.1 Logistic Regression

This model appears to have gained some insight in the data and accurately defined a majority of the data. The accuracy of the model is >95% which indicates that it was able to determine a trend and apply it in a useful manner in the predictions during evaluation. Further, the confusion matrix further supports the high accuracy and likely usefulness of the model with only 3 false assignments. However, in analysis this is only to determine if there is a correlation between binary assignment and the emission strength x error in measurement. This doesn't aid us in our overall randomness determination, but it does determine that uncertainty has a role in the binary assignment and the overall trust of emission strength.

4.3.2 Bidirectional LSTM

this model has a varied accuracy according the MSE, MAE and RMSE metrics we can interpret that the line is close to the actual values. But the R^2 indicates that there is no analysis or accounting for the explained variation of the data indicating a non-linear relationship. This means that the model could be error prone or requiring further optimisation.

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.

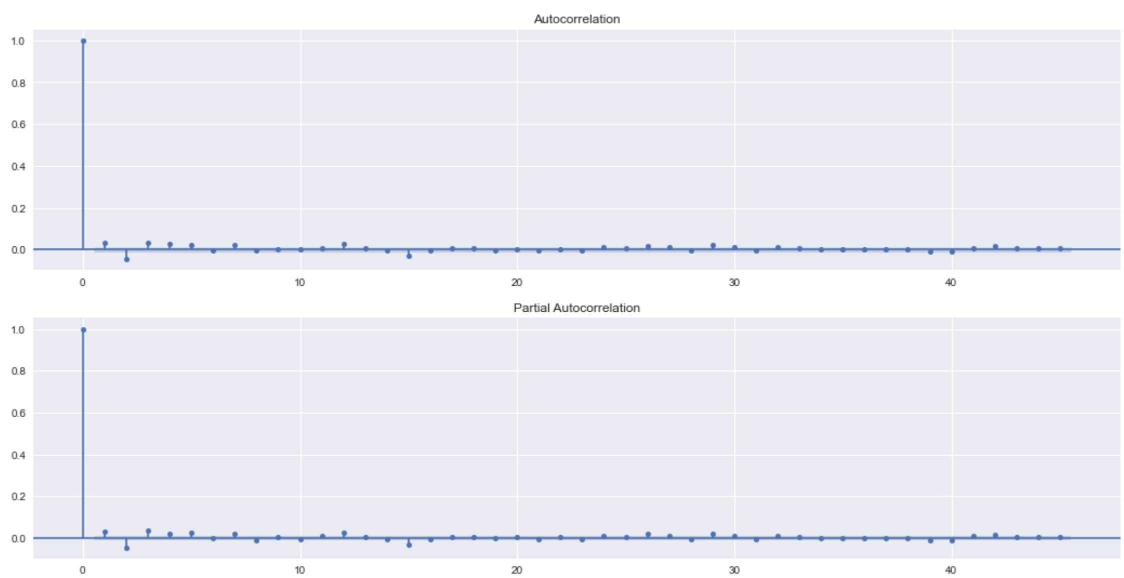
6 FUNCTION CODE FOR RUNS TEST

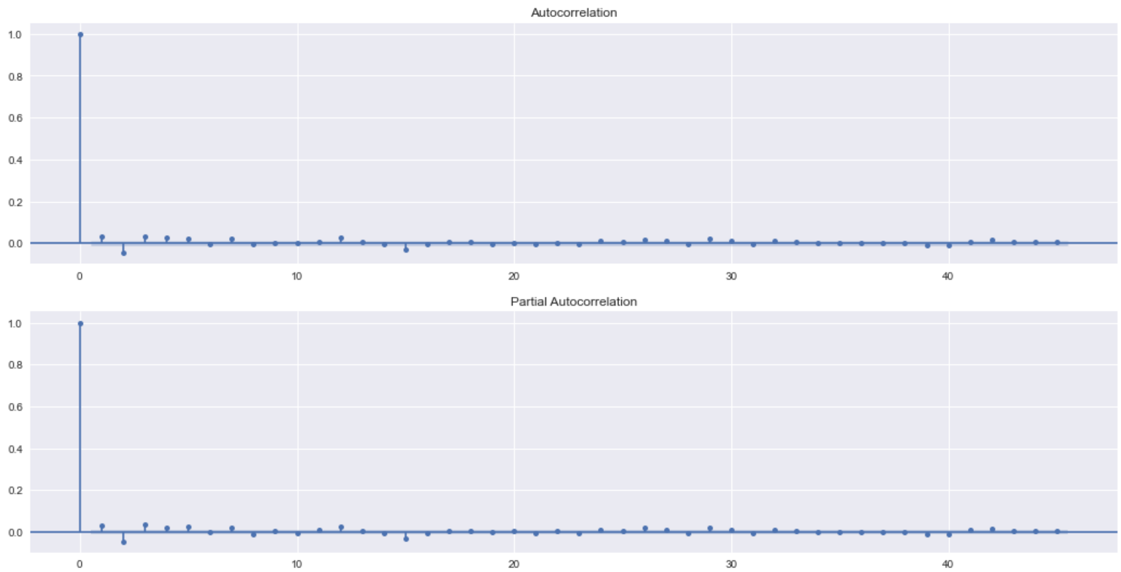
7 Below we begin autocorrelation and autocovariance analysis

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\oxlay\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([ 1.00000000e+00,  3.02297520e-02, -4.45596588e-02,  3.11934730e-02,
        2.59222929e-02,  2.29438690e-02, -2.72483478e-03,  2.17140483e-02,
       -5.63815508e-03,  1.61127481e-03, -7.10167005e-04])
```

[ ]:
```
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	0.598393	NaN	NaN	NaN	NaN	NaN	NaN
1	0.590859	0.598393	NaN	NaN	NaN	NaN	NaN
2	0.449643	0.590859	0.598393	NaN	NaN	NaN	NaN
3	0.682860	0.449643	0.590859	0.598393	NaN	NaN	NaN
4	0.490026	0.682860	0.449643	0.590859	0.598393	NaN	NaN
…	…	…	…	…	…	…	…
26995	0.539079	0.396929	1.014446	0.659313	1.173766	0.606806	0.500412
26996	0.324070	0.539079	0.396929	1.014446	0.659313	1.173766	0.606806
26997	0.291341	0.324070	0.539079	0.396929	1.014446	0.659313	1.173766

```
26998  0.346267  0.291341  0.324070  0.539079  0.396929  1.014446  0.659313
26999  0.513315  0.346267  0.291341  0.324070  0.539079  0.396929  1.014446

        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
…             …         …         …         …
26995  0.409631  0.698172  0.262350  0.447577
26996  0.500412  0.409631  0.698172  0.262350
26997  0.606806  0.500412  0.409631  0.698172
26998  1.173766  0.606806  0.500412  0.409631
26999  0.659313  1.173766  0.606806  0.500412

[27000 rows x 11 columns]
```

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

[ ]:
```
array([ 1.00000000e+00,  3.02297663e-02, -4.45598682e-02,  3.11938480e-02,
        2.59227920e-02,  2.29443159e-02, -2.72489307e-03,  2.17154085e-02,
       -5.63853223e-03,  1.61145686e-03, -7.10263111e-04])
```

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

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

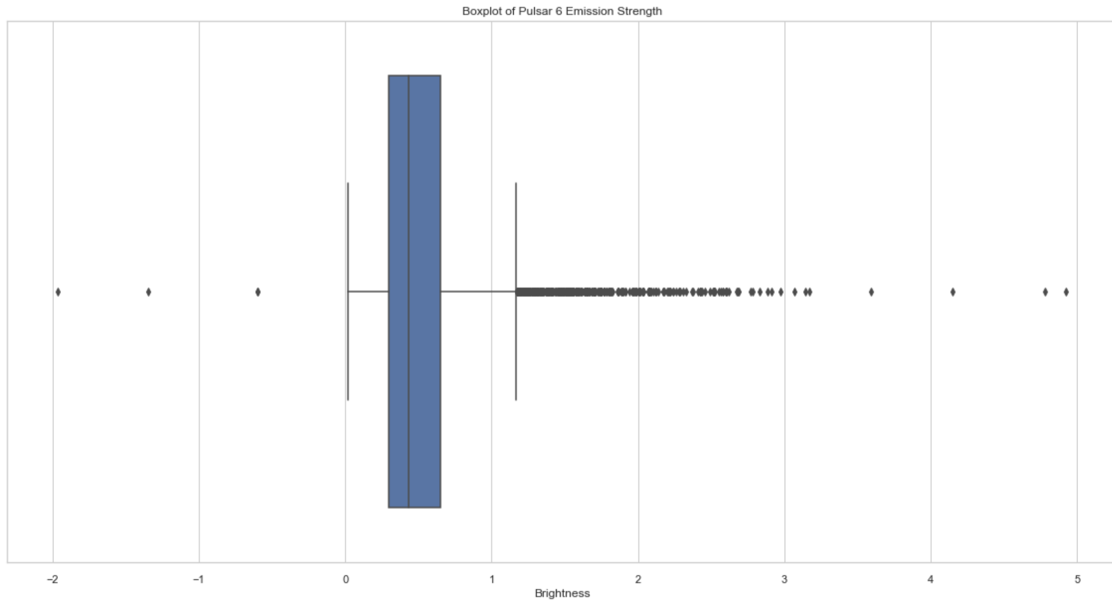
[ ]:
```
       Pulse Number  Brightness  Uncertainty  Binary  \
0                 1    0.598393     0.056431       1
5                 6    0.586071     0.052649       1
10               11    0.670907     0.049539       1
15               16    0.404870     0.056679       0
20               21    0.297223     0.048826       0
…               …         …           …       …
26975         26976    0.384184     0.070075       0
26980         26981    0.317133     0.055033       0
26985         26986    0.447577     0.054011       1
26990         26991    0.606806     0.043464       1
26995         26996    0.539079     0.063854       1

       RollingMeanEmissions5ths  RollingMeanEmissions10ths  \
```

```
0                        NaN                        NaN
5                   0.559892                        NaN
10                  0.526181                   0.543036
15                  0.454187                   0.490184
20                  0.350959                   0.402573
…                        …                         …
26975               0.595775                   0.483936
26980               0.589773                   0.592774
26985               0.440076                   0.514925
26990               0.495474                   0.467775
26995               0.756707                   0.626091

       RollingMedianEmissions5ths  RollingMedianEmissions10ths
0                             NaN                          NaN
5                        0.586071                          NaN
10                       0.429094                     0.538048
15                       0.465406                     0.447250
20                       0.297223                     0.414102
…                             …                            …
26975                    0.516402                     0.474701
26980                    0.317133                     0.474701
26985                    0.447577                     0.382355
26990                    0.500412                     0.450191
26995                    0.659313                     0.572943

[5400 rows x 8 columns]
```

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

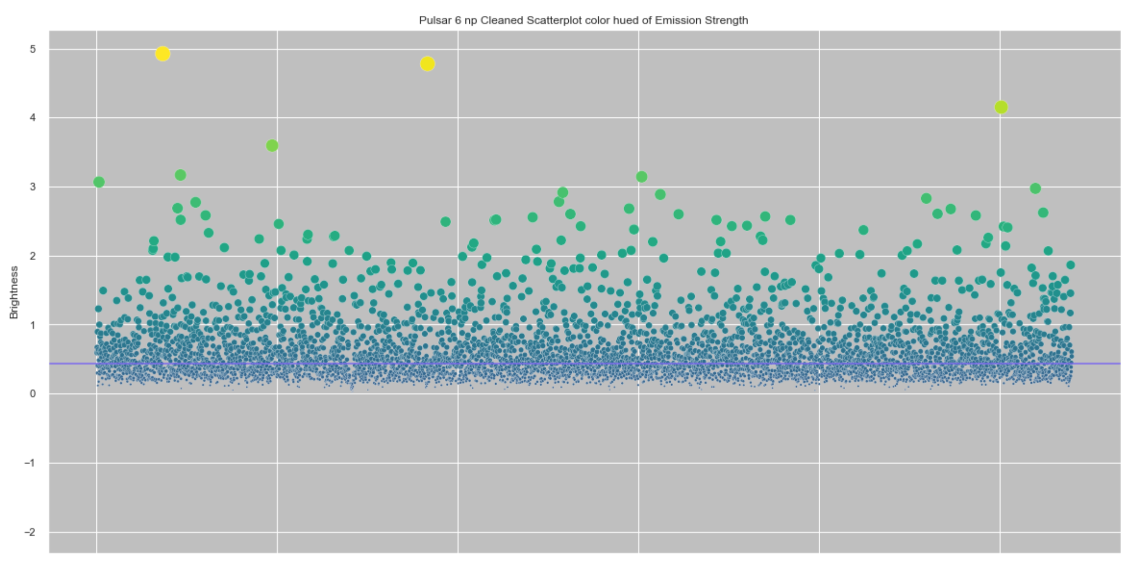
[ ]:
```
0.43021975
```

[ ]:
```
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=0.43021975, ls='-',c='mediumslateblue')
```

```
c:\Users\oxlay\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=0.6508051, 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 > 0.43021975)]))
     print(len(held5ths[(held5ths.Brightness < 0.43021975)]))
```

```
2700
2700
```

Randomness testing

```
[ ]: np.savetxt(r'every5thbinarypulsar1.txt', held5ths.Binary, fmt='%d',␣
      ↪delimiter='')
     np.savetxt(r'allpulsar1.txt', pulsar.Binary, fmt='%d', delimiter='')
```

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

```
[ ]: 0        1
     1        1
     2        1
     3        1
     4        1
             ..
     26995    1
     26996    0
     26997    0
     26998    0
     26999    1
     Name: Binary, Length: 27000, dtype: int32
```