pulsar3

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/J0835-4510.pulses", sep = ' ', header = None, names␣
↪= colnames)
```

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

[ ]:
```
(1331, 3)
```

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

[ ]:

	Pulse Number	Brightness	Uncertainty
0	1	0.984043	0.053831
1	2	2.487928	0.048796
2	3	1.690295	0.025639
3	4	1.196142	0.039539
4	5	1.979783	0.041460
5	6	2.297645	0.054210
6	7	2.322135	0.043554
7	8	2.289047	0.049957
8	9	2.442574	0.025110
9	10	2.136332	0.022712
10	11	1.976790	0.037551
11	12	2.445764	0.047004
12	13	1.937017	0.028561
13	14	2.315184	0.045216
14	15	2.584888	0.040232
15	16	1.581452	0.030372
16	17	1.849656	0.024236
17	18	2.529834	0.048330
18	19	2.894401	0.066794

```
19            20    2.769474     0.059082
20            21    1.824490     0.036531
21            22    1.498133     0.035557
22            23    2.005834     0.028621
23            24    2.594836     0.032925
24            25    2.745045     0.055348
```

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

[ ]:
```
       Pulse Number   Brightness  Uncertainty
count   1331.000000  1331.000000  1331.000000
mean     666.000000     2.248107     0.039495
std      384.370915     0.591161     0.013056
min        1.000000     0.633413     0.012888
25%      333.500000     1.825375     0.030223
50%      666.000000     2.255182     0.037513
75%      998.500000     2.682259     0.046771
max     1331.000000     4.050718     0.098902
```

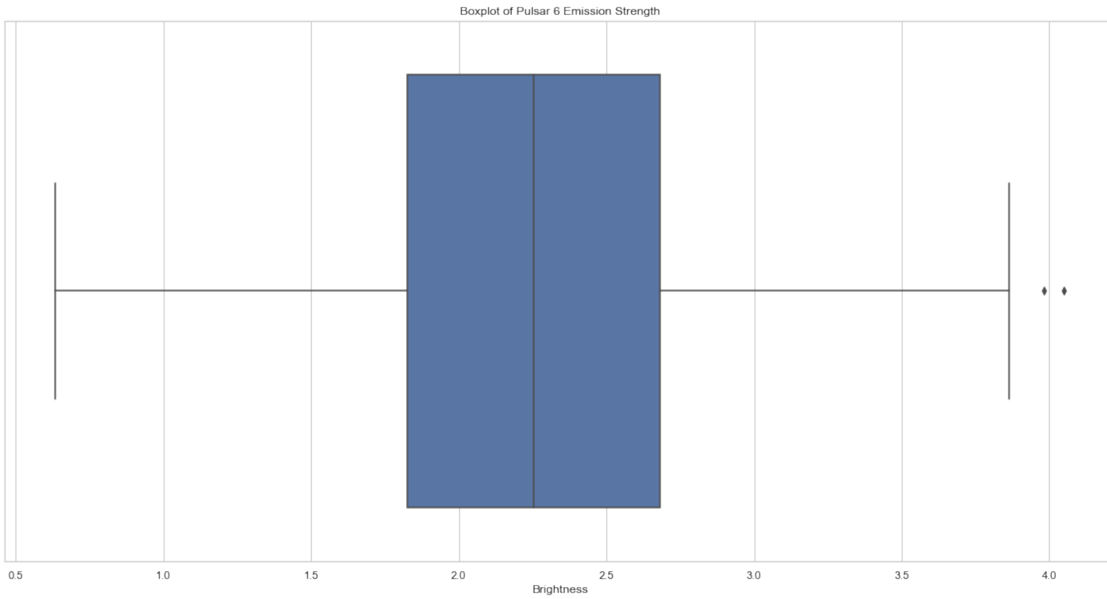
[ ]:
```
nullBoolBrightness = pd.isnull(pulsar["Brightness"])

pulsar[nullBoolBrightness]
```

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

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

[ ]:
```
count    1331.000000
mean        2.248107
std         0.591161
min         0.633413
25%         1.825375
50%         2.255182
75%         2.682259
max         4.050718
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:  2.255182
```

[ ]:
```
pulsar
```

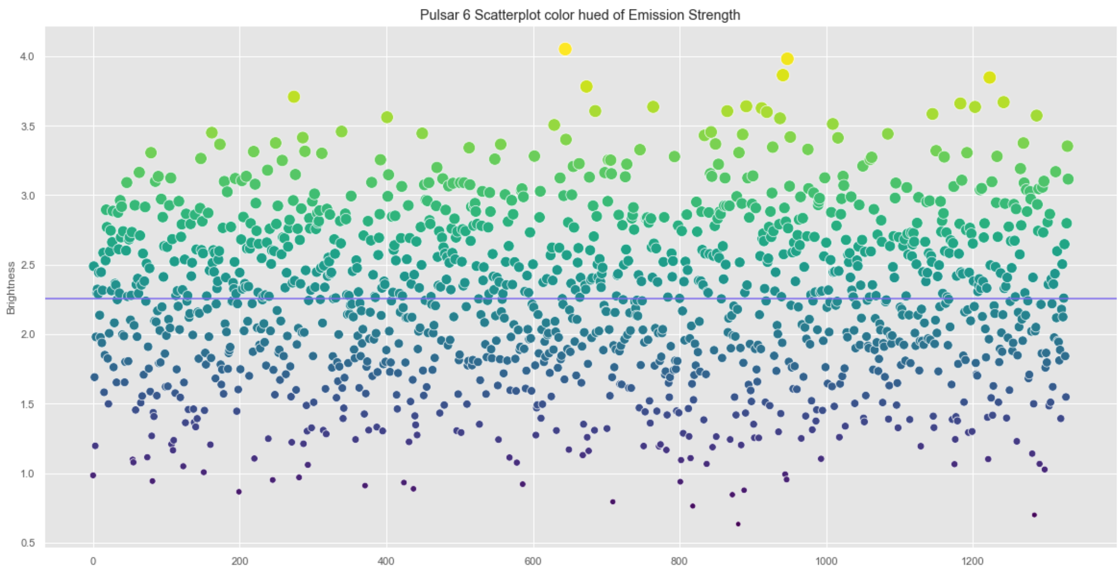
[ ]:

	Pulse Number	Brightness	Uncertainty	Binary
0	1	0.984043	0.053831	0
1	2	2.487928	0.048796	1
2	3	1.690295	0.025639	0
3	4	1.196142	0.039539	0
4	5	1.979783	0.041460	0
…	…	…	…	…
1326	1327	1.842016	0.028216	0
1327	1328	1.547695	0.024030	0
1328	1329	2.797312	0.035090	1
1329	1330	3.351977	0.052178	1
1330	1331	3.115255	0.035134	1

[1331 rows x 4 columns]

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

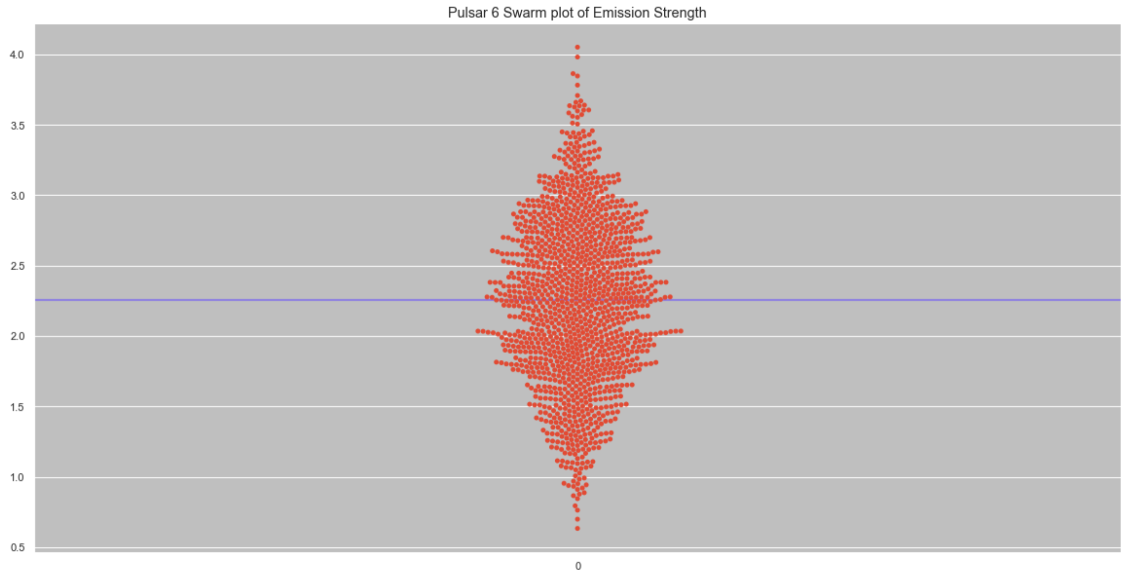
```
ax = sns.scatterplot(data=pulsar["Brightness"], s= strength*50, c=strength,␣
↪cmap="viridis", marker="o").set_title('Pulsar 6 Scatterplot color hued of␣
↪Emission Strength')
ax= plt.axhline( y=2.255182, ls='-',c='mediumslateblue')
```



[ ]:
```
print(len(pulsar[(pulsar.Brightness > 2.255182)]))
print(len(pulsar[(pulsar.Brightness < 2.255182)]))
```

665
665

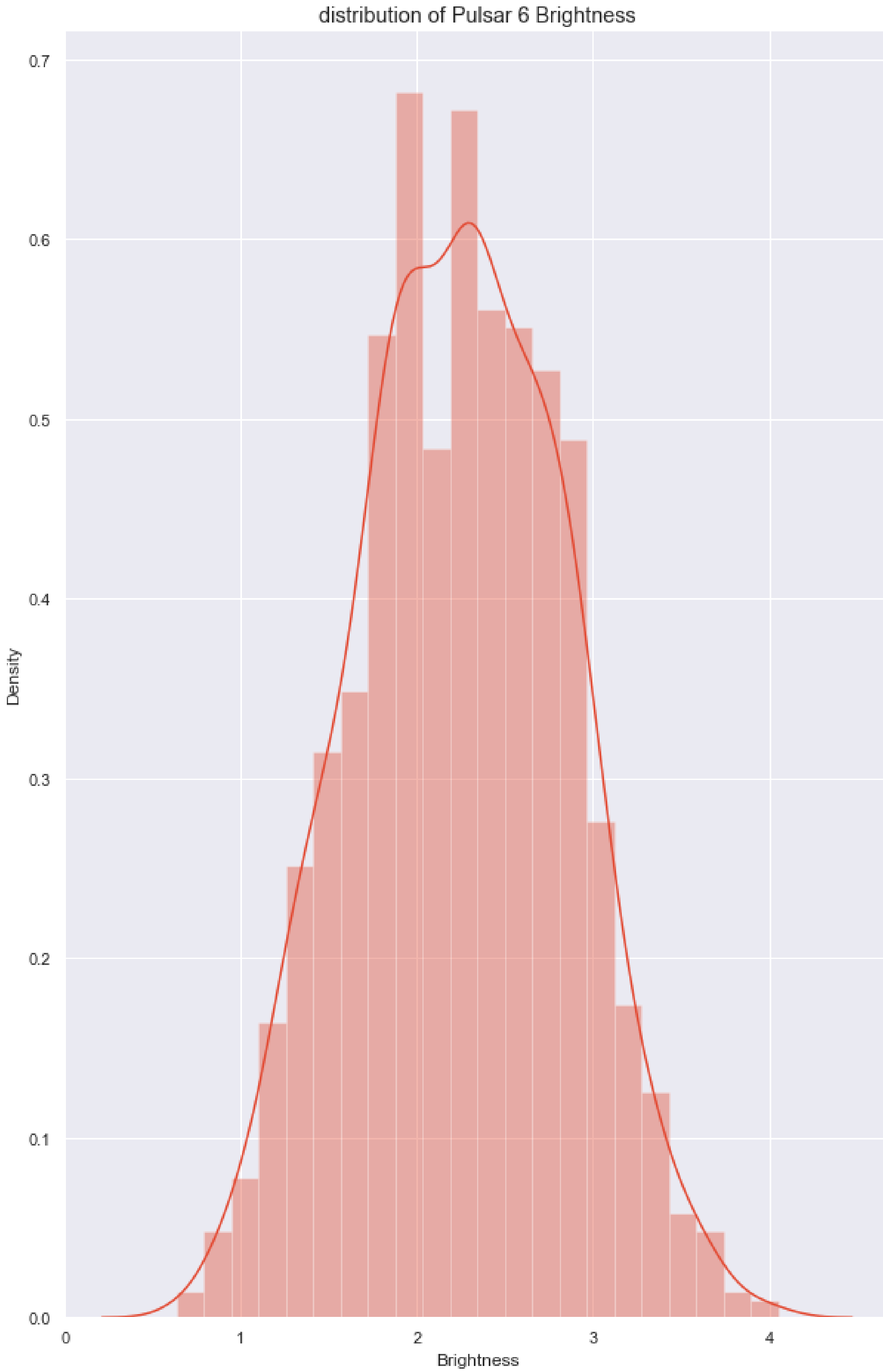
[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = pulsar.Brightness.values
ax = plt.axhline( y=2.255182, ls='-',c='mediumslateblue')
ax = sns.swarmplot(data=pulsar["Brightness"], c="blue").set_title('Pulsar 6␣
↪Swarm plot of Emission Strength')
```




```
[ ]: 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')
```

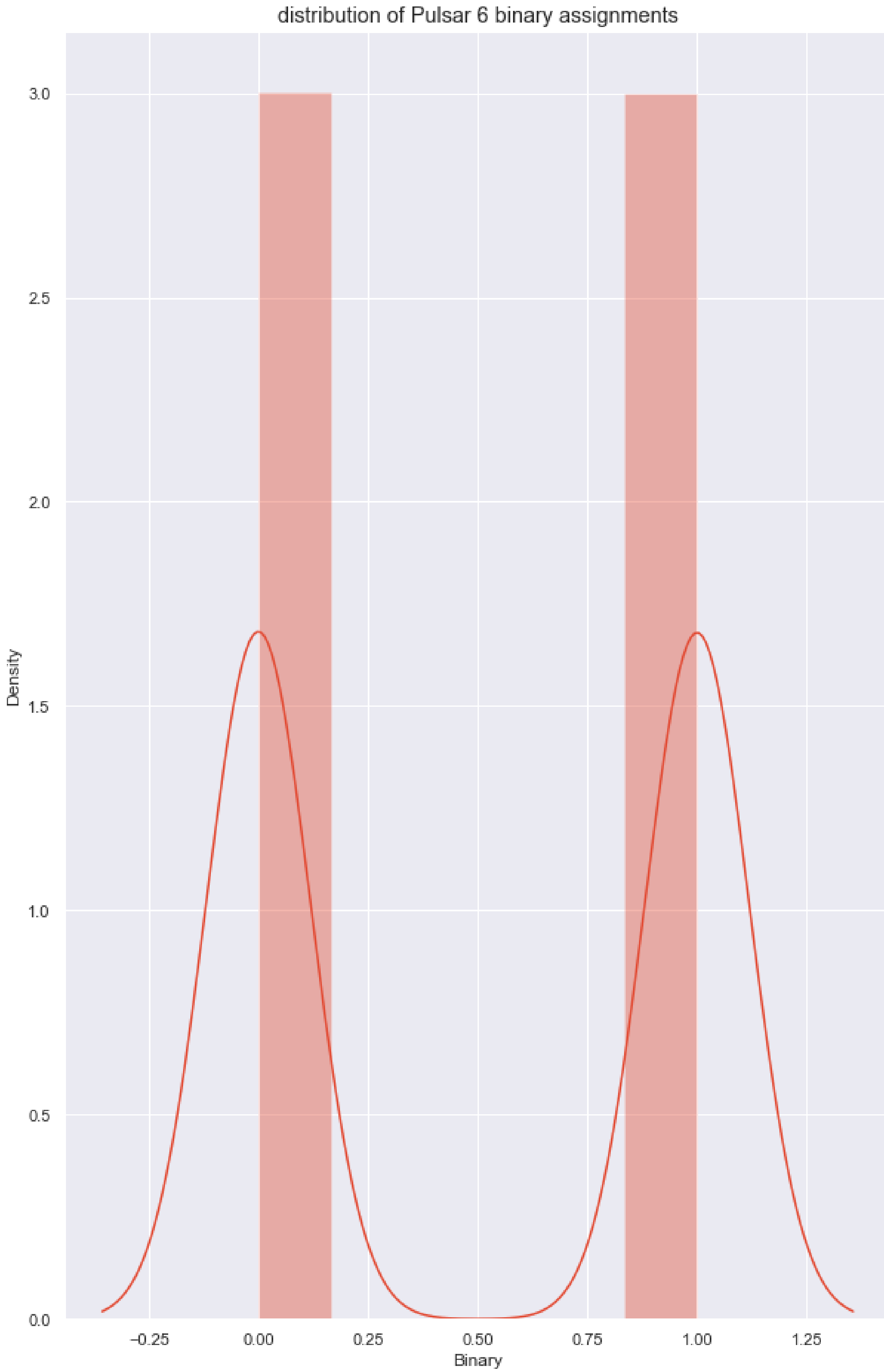


distribution of Pulsar 6 Brightness
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
0
1
2
3
4
Brightness

[ ]:
```
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')
```

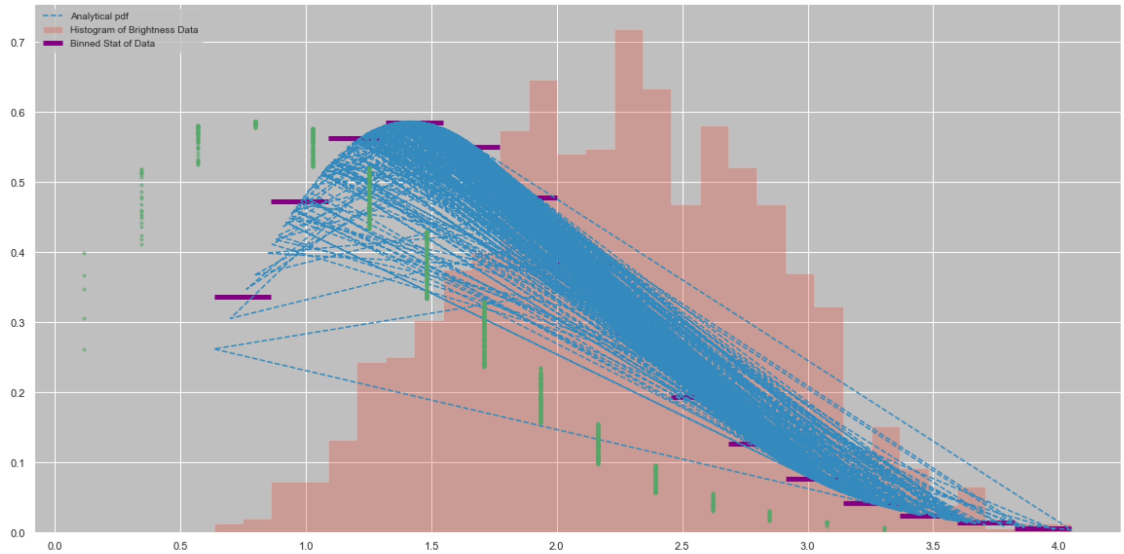

distribution of Pulsar 6 binary assignments
Density
Binary
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25

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

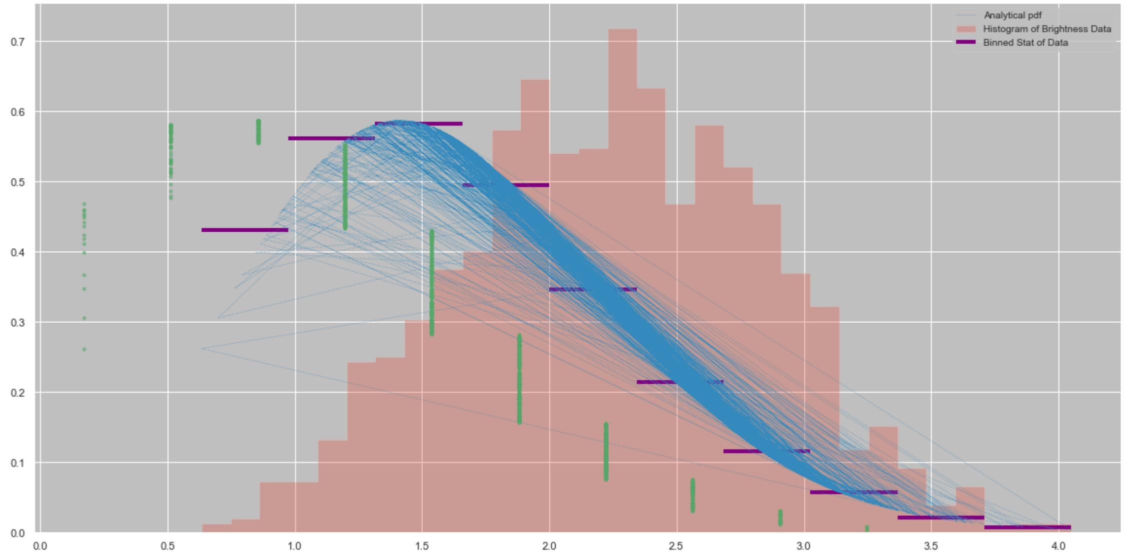
[ ]:
```
data = pulsar["Brightness"]
data
```

[ ]:
```
0       0.984043
1       2.487928
2       1.690295
3       1.196142
4       1.979783
          …
1326    1.842016
1327    1.547695
1328    2.797312
1329    3.351977
1330    3.115255
Name: Brightness, Length: 1331, dtype: float64
```

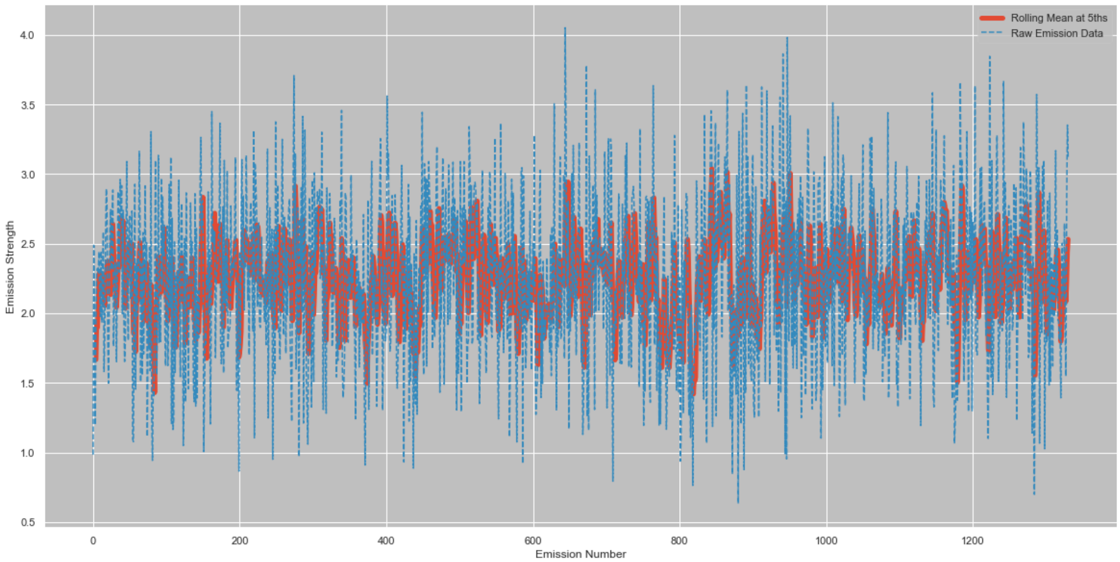
[ ]:
```
dataPDF = stats.maxwell.pdf(data)
bin_means, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='mean', bins=15)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

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()
```



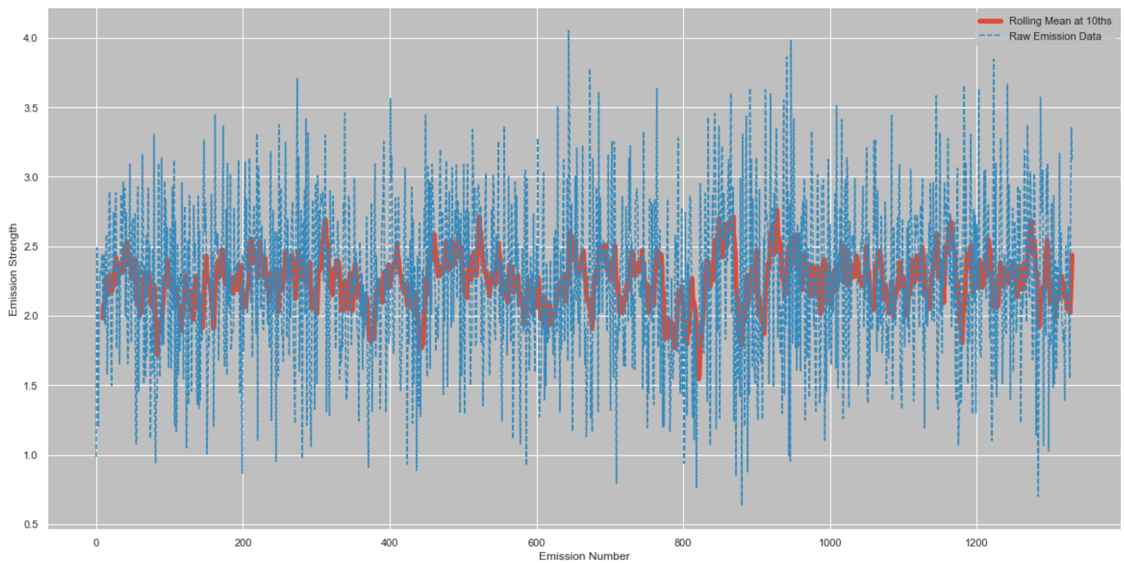
[ ]:
```
bin_median, bin_edges, binnumber = stats.binned_statistic(data, dataPDF,
        statistic='median', bins=10)
bin_width = (bin_edges[1] - bin_edges[0])
bin_centers = bin_edges[1:] - bin_width/2

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



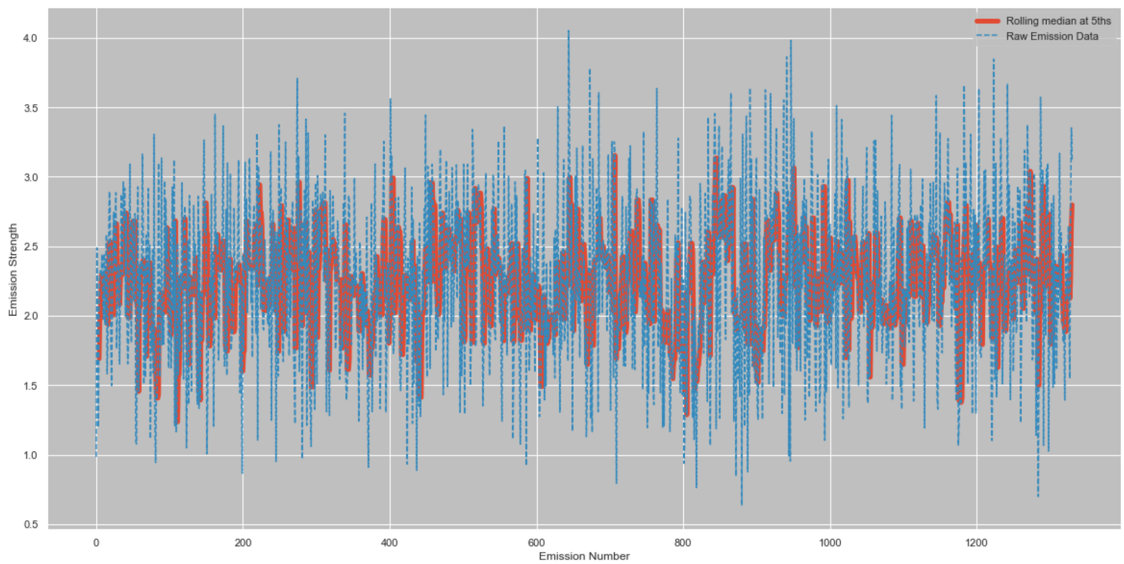
[ ]:
```python
pulsar['RollingMeanEmissions5ths'] = pulsar["Brightness"].rolling(5).mean()

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



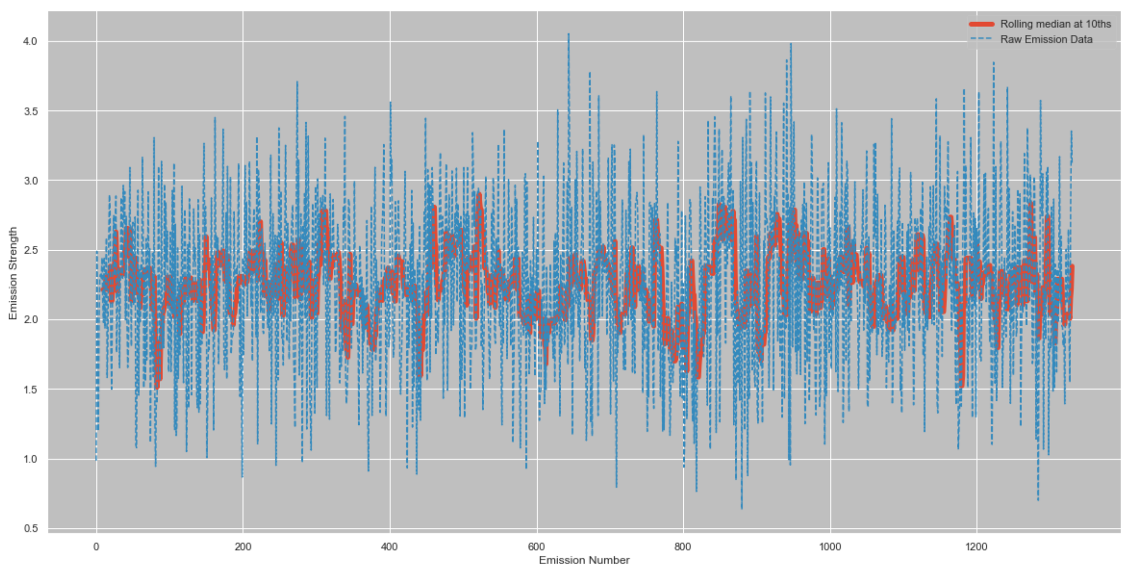
[ ]:
```
pulsar['RollingMeanEmissions10ths'] = pulsar["Brightness"].rolling(10).mean()

plt.figure(figsize=(20,10))
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.figure(figsize=(20,10))
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.figure(figsize=(20,10))
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    0.984043     0.053831       0                       NaN
1              2    2.487928     0.048796       1                       NaN
2              3    1.690295     0.025639       0                       NaN
3              4    1.196142     0.039539       0                       NaN
4              5    1.979783     0.041460       0                  1.667638
5              6    2.297645     0.054210       1                  1.930359
6              7    2.322135     0.043554       1                  1.897200
7              8    2.289047     0.049957       1                  2.016950
8              9    2.442574     0.025110       1                  2.266237
9             10    2.136332     0.022712       0                  2.297547
10            11    1.976790     0.037551       0                  2.233376
11            12    2.445764     0.047004       1                  2.258101
12            13    1.937017     0.028561       0                  2.187695
13            14    2.315184     0.045216       1                  2.162217
14            15    2.584888     0.040232       1                  2.251929
15            16    1.581452     0.030372       0                  2.172861
16            17    1.849656     0.024236       0                  2.053639
17            18    2.529834     0.048330       1                  2.172203
18            19    2.894401     0.066794       1                  2.288046
19            20    2.769474     0.059082       1                  2.324963
20            21    1.824490     0.036531       0                  2.373571
21            22    1.498133     0.035557       0                  2.303266
22            23    2.005834     0.028621       0                  2.198466
23            24    2.594836     0.032925       1                  2.138553
24            25    2.745045     0.055348       1                  2.133668

    RollingMeanEmissions10ths  RollingMedianEmissions5ths  \
0                         NaN                         NaN
1                         NaN                         NaN
2                         NaN                         NaN
3                         NaN                         NaN
4                         NaN                    1.690295
5                         NaN                    1.979783
6                         NaN                    1.979783
7                         NaN                    2.289047
8                         NaN                    2.297645
9                    1.982592                    2.297645
10                   2.081867                    2.289047
11                   2.077651                    2.289047
12                   2.102323                    2.136332
13                   2.214227                    2.136332
```

```
14                    2.274738                     2.315184
15                    2.203118                     2.315184
16                    2.155870                     1.937017
17                    2.179949                     2.315184
18                    2.225132                     2.529834
19                    2.288446                     2.529834
20                    2.273216                     2.529834
21                    2.178453                     2.529834
22                    2.185335                     2.005834
23                    2.213300                     2.005834
24                    2.229315                     2.005834

    RollingMedianEmissions10ths
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN
5                           NaN
6                           NaN
7                           NaN
8                           NaN
9                      2.212689
10                     2.212689
11                     2.212689
12                     2.212689
13                     2.293346
14                     2.306414
15                     2.302116
16                     2.212689
17                     2.225758
18                     2.225758
19                     2.380474
20                     2.380474
21                     2.126100
22                     2.160509
23                     2.267834
24                     2.267834
```

4.1 Binary Classification

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

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

[ ]:
```
   Brightness  Uncertainty
0    0.984043     0.053831
1    2.487928     0.048796
2    1.690295     0.025639
3    1.196142     0.039539
4    1.979783     0.041460
```

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

[ ]:
```
0    0
1    1
2    0
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)  =  135
```

```
False Positive(FP) =  0
True Negative(TN)  =  131
False Negative(FN) =  1
```

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

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

```
Accuracy of the model is  0.9962546816479401
```

4.2 Bidirectional LSTM Model

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

[ ]:
```
[[0.9840433, 0.05383067],
 [2.487928, 0.04879624],
 [1.690295, 0.02563856],
 [1.196142, 0.03953864],
 [1.979783, 0.04146007],
 [2.297645, 0.05420977],
 [2.322135, 0.043554],
 [2.289047, 0.04995664],
 [2.442574, 0.02511043],
 [2.136332, 0.0227117]]
```

[ ]:
```
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
```

[ ]: (984, 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
52/52 [==============================] - 3s 28ms/step - loss: 0.2008 - accuracy:
0.0000e+00
Epoch 2/50
52/52 [==============================] - 1s 28ms/step - loss: 0.0020 - accuracy:
0.0000e+00
Epoch 3/50
52/52 [==============================] - 2s 33ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 4/50
52/52 [==============================] - 2s 32ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 5/50
52/52 [==============================] - 2s 36ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 6/50
52/52 [==============================] - 2s 40ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 7/50
52/52 [==============================] - 2s 34ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 8/50
52/52 [==============================] - 2s 41ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 9/50
```

```
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 10/50
52/52 [==============================] - 3s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 11/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 12/50
52/52 [==============================] - 3s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 13/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 14/50
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 15/50
52/52 [==============================] - 3s 51ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 16/50
52/52 [==============================] - 2s 46ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 17/50
52/52 [==============================] - 2s 41ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 18/50
52/52 [==============================] - 3s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 19/50
52/52 [==============================] - 3s 51ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 20/50
52/52 [==============================] - 2s 40ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 21/50
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 22/50
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 23/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 24/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 25/50
```

```
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 26/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 27/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 28/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 29/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 30/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 31/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 32/50
52/52 [==============================] - 2s 45ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 33/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 34/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 35/50
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 36/50
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 37/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 38/50
52/52 [==============================] - 2s 40ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 39/50
52/52 [==============================] - 3s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 40/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 41/50
```

```
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 42/50
52/52 [==============================] - 2s 45ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 43/50
52/52 [==============================] - 2s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 44/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 45/50
52/52 [==============================] - 3s 48ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 46/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 47/50
52/52 [==============================] - 2s 47ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 48/50
52/52 [==============================] - 2s 44ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 49/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - accuracy:
0.0000e+00
Epoch 50/50
52/52 [==============================] - 3s 49ms/step - loss: 0.0018 - 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([[0.99977815],
       [0.999779  ],
       [0.99977994],
       [0.99978054],
       [0.99977887],
       [0.9997796 ],
       [0.9997798 ],
       [0.9997805 ],
       [0.9997796 ],
       [0.9997789 ]], dtype=float32)
```

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

```
8/8 [==============================] - 1s 15ms/step - loss: 0.0018 - accuracy:
0.0000e+00
```

[ ]:
```
[0.001771298935636878, 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, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0,
1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
```

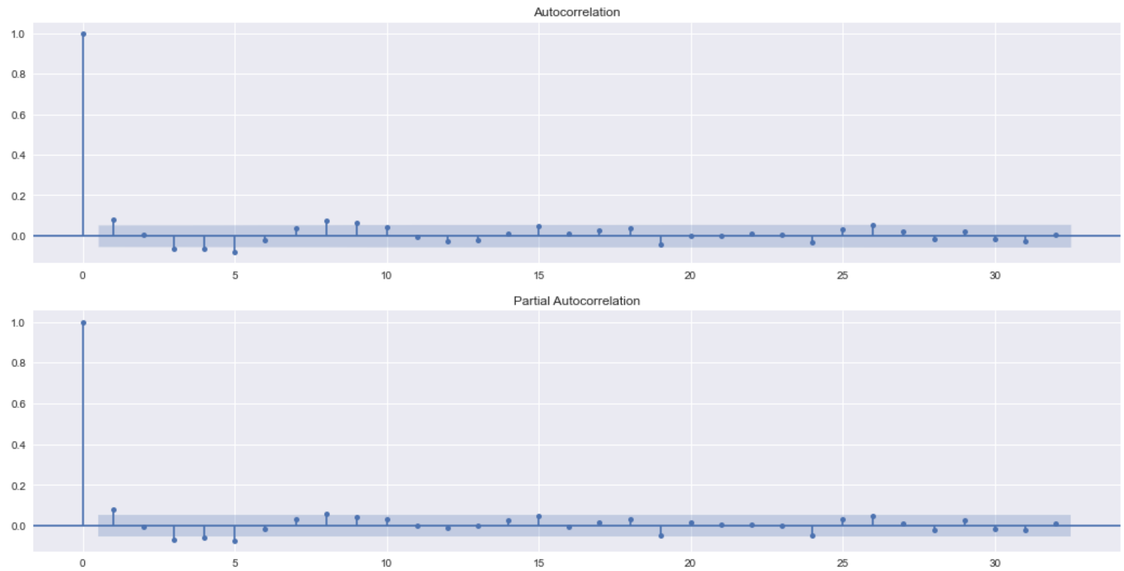
```
1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,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

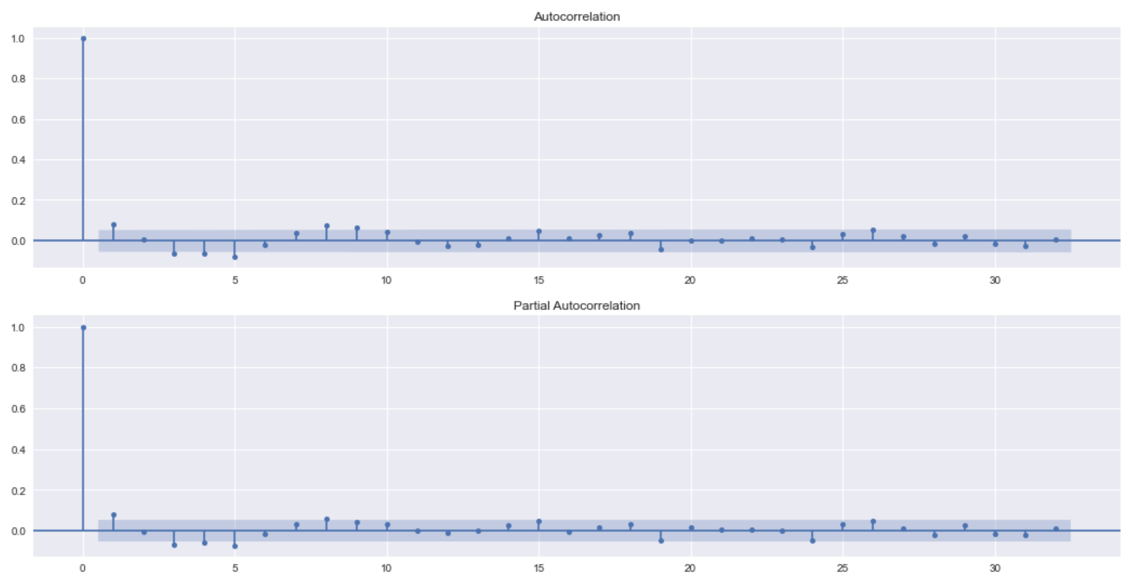
7 Below we begin autocorrelation and autocovariance analysis

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.        ,  0.08095517,  0.00412699, -0.06721578, -0.06554554,
            -0.07977771, -0.02053543,  0.03766276,  0.07611705,  0.06106126,
             0.04441705])
```

```
[ ]: 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     0.984043       NaN       NaN       NaN       NaN       NaN       NaN
     1     2.487928  0.984043       NaN       NaN       NaN       NaN       NaN
     2     1.690295  2.487928  0.984043       NaN       NaN       NaN       NaN
     3     1.196142  1.690295  2.487928  0.984043       NaN       NaN       NaN
     4     1.979783  1.196142  1.690295  2.487928  0.984043       NaN       NaN
     …          …         …         …         …         …         …         …
     1326  1.842016  2.646750  2.258860  2.123736  2.503202  2.178636  1.392491
     1327  1.547695  1.842016  2.646750  2.258860  2.123736  2.503202  2.178636
     1328  2.797312  1.547695  1.842016  2.646750  2.258860  2.123736  2.503202
     1329  3.351977  2.797312  1.547695  1.842016  2.646750  2.258860  2.123736
     1330  3.115255  3.351977  2.797312  1.547695  1.842016  2.646750  2.258860

           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
     …          …         …         …         …
     1326  1.886326  1.810641  1.943447  1.950708
     1327  1.392491  1.886326  1.810641  1.943447
     1328  2.178636  1.392491  1.886326  1.810641
     1329  2.503202  2.178636  1.392491  1.886326
     1330  2.123736  2.503202  2.178636  1.392491

     [1331 rows x 11 columns]
```

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

[ ]:
```
array([ 1.        ,  0.0811623 ,  0.00414645, -0.06751767, -0.06595236,
       -0.08029629, -0.02066581,  0.0379259 ,  0.07664111,  0.06149054,
        0.04473245])
```

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
```

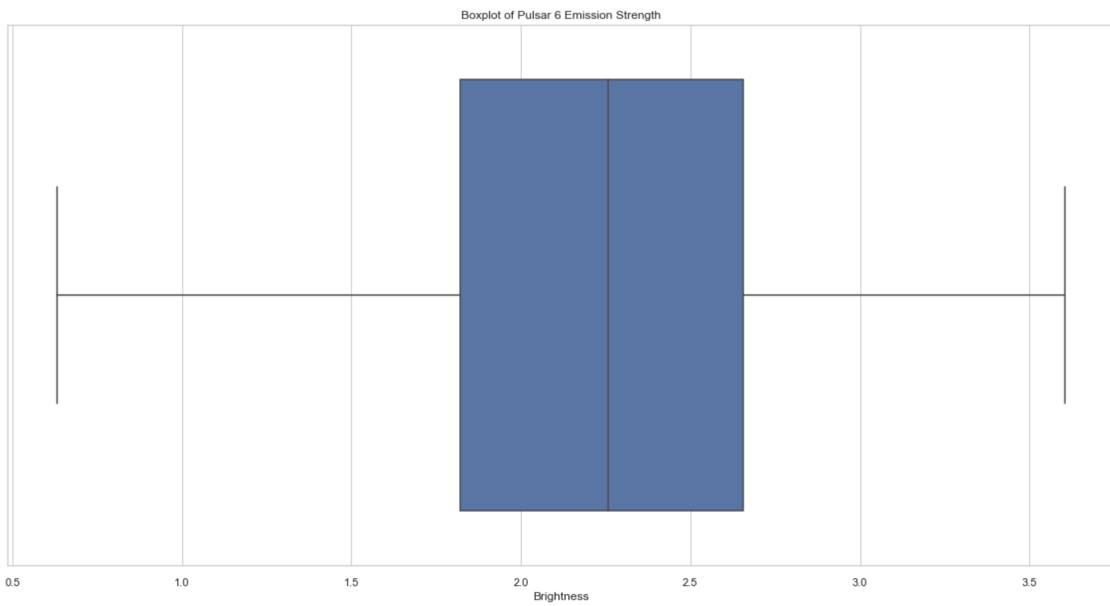
[ ]:
```
      Pulse Number  Brightness  Uncertainty  Binary  RollingMeanEmissions5ths  \
0                1    0.984043     0.053831       0                       NaN
5                6    2.297645     0.054210       1                  1.930359
10              11    1.976790     0.037551       0                  2.233376
15              16    1.581452     0.030372       0                  2.172861
20              21    1.824490     0.036531       0                  2.373571
…              …         …          …      …                       …
1310          1311    2.360064     0.034759       1                  2.020855
1315          1316    2.596850     0.048041       1                  2.458422
1320          1321    1.392491     0.030957       0                  1.796723
1325          1326    2.646750     0.036691       1                  2.342237
1330          1331    3.115255     0.035134       1                  2.530851

      RollingMeanEmissions10ths  RollingMedianEmissions5ths  \
0                           NaN                         NaN
5                           NaN                    1.979783
10                     2.081867                    2.289047
15                     2.203118                    2.315184
20                     2.273216                    2.529834
…                          …                          …
1310                   2.141252                    2.218181
1315                   2.239639                    2.434470
1320                   2.127572                    1.886326
1325                   2.069480                    2.258860
1330                   2.436544                    2.797312

      RollingMedianEmissions10ths
0                             NaN
5                             NaN
10                       2.212689
15                       2.302116
20                       2.380474
```

```
…                         …
1310                      2.287940
1315                      2.289122
1320                      1.962875
1325                      2.037222
1330                      2.381031

[267 rows x 8 columns]
```

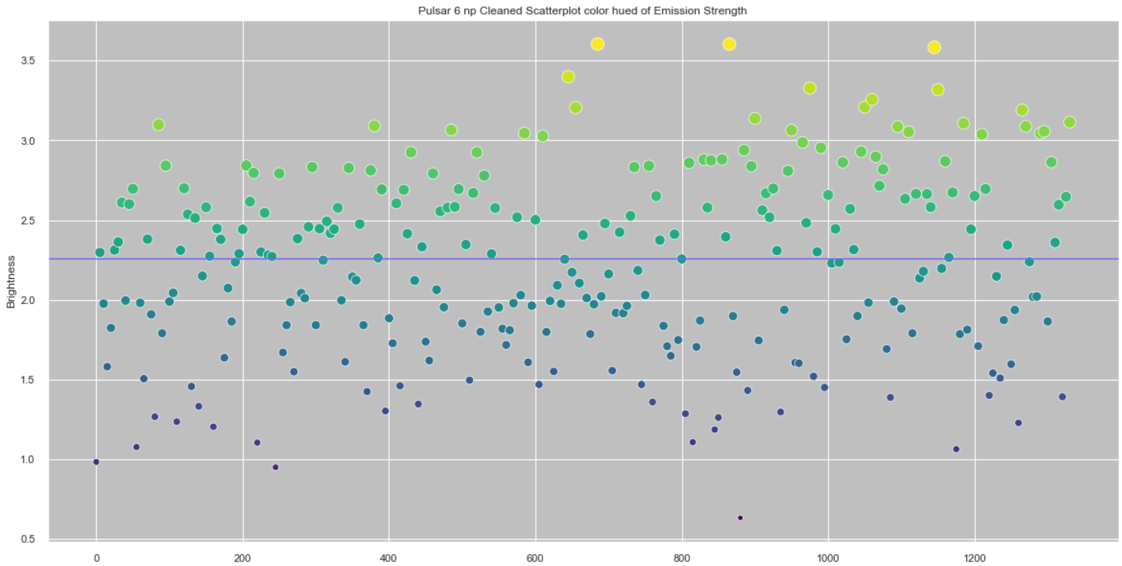
[ ]:
```python
medianheld5ths = held5ths["Brightness"].median()
medianheld5ths
```

[ ]: 2.256816

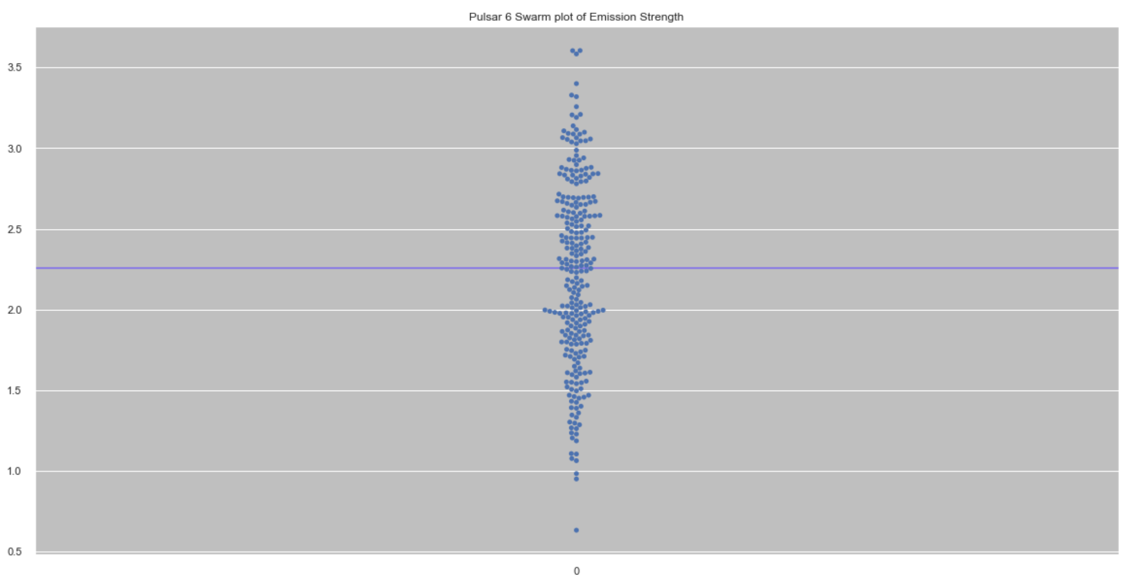
[ ]:
```python
plt.figure(figsize=(20,10))
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=held5ths["Brightness"]).set_title("Boxplot of Pulsar 6␣
↪Emission Strength")
```



[ ]:
```python
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=2.256816, ls='-',c='mediumslateblue')
```



[ ]:
```
plt.figure(figsize=(20,10))
sns.set_style("darkgrid", {"axes.facecolor": ".75"})
strength = held5ths.Brightness.values
ax = plt.axhline( y=2.256816, 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 > 2.256816)]))
print(len(held5ths[(held5ths.Brightness < 2.256816)]))
```

```
133
133
```

Randomness testing

[ ]:
```
np.savetxt(r'every5thbinarypulsar3.txt', held5ths.Binary, fmt='%d',␣
 ↪delimiter='')
np.savetxt(r'allpulsar3.txt', pulsar.Binary, fmt='%d', delimiter='')
```

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

[ ]:
```
0       0
1       1
2       0
3       0
4       0
       ..
1326    0
1327    0
1328    1
1329    1
1330    1
Name: Binary, Length: 1331, dtype: int32
```