pulsar5

October 8, 2022

1 Pulsar Emission Data Analysis

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

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

3 Section for extracting from a tar file.

Currently implemented for original TAR File structure.

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

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

3.1 Beginning of Exploration

3.1.1 Examining the data

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

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

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

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

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

[ ]:

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

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

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

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

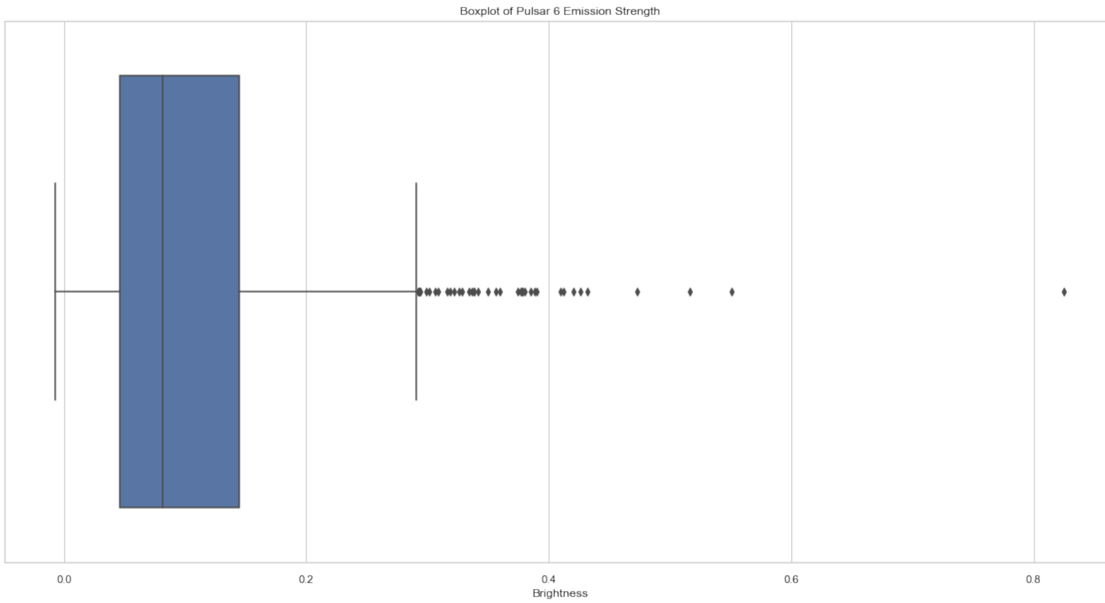
[ ]:
```
nullBoolBrightness = pd.isnull(pulsar["Brightness"])

pulsar[nullBoolBrightness]
```

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

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

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

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




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

```
Median of Pulsar6:  0.081228
```

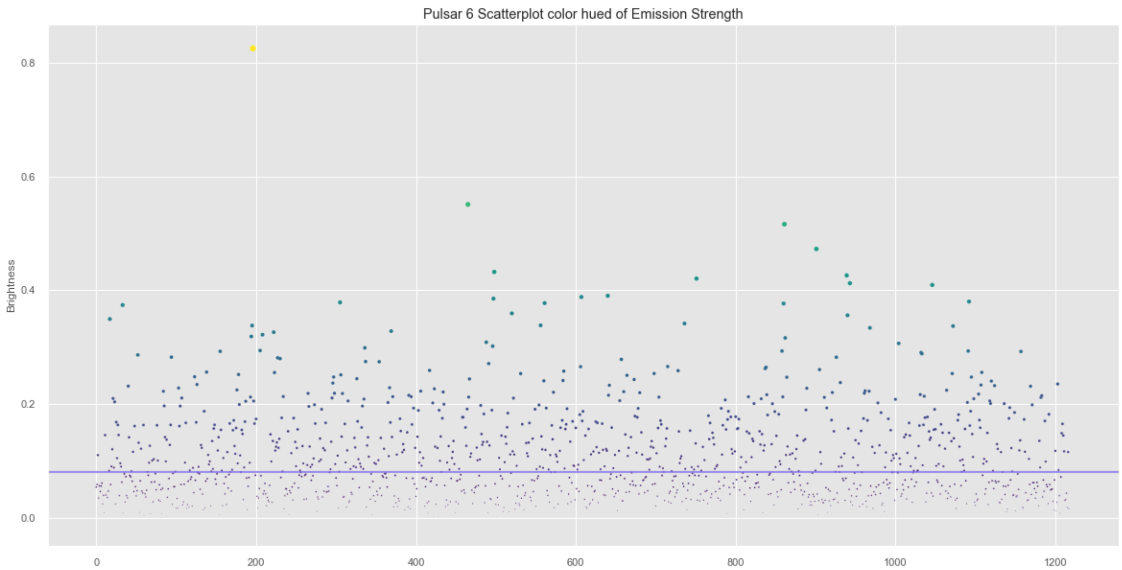
```
[ ]: pulsar
```

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

     [1219 rows x 4 columns]
```

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

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

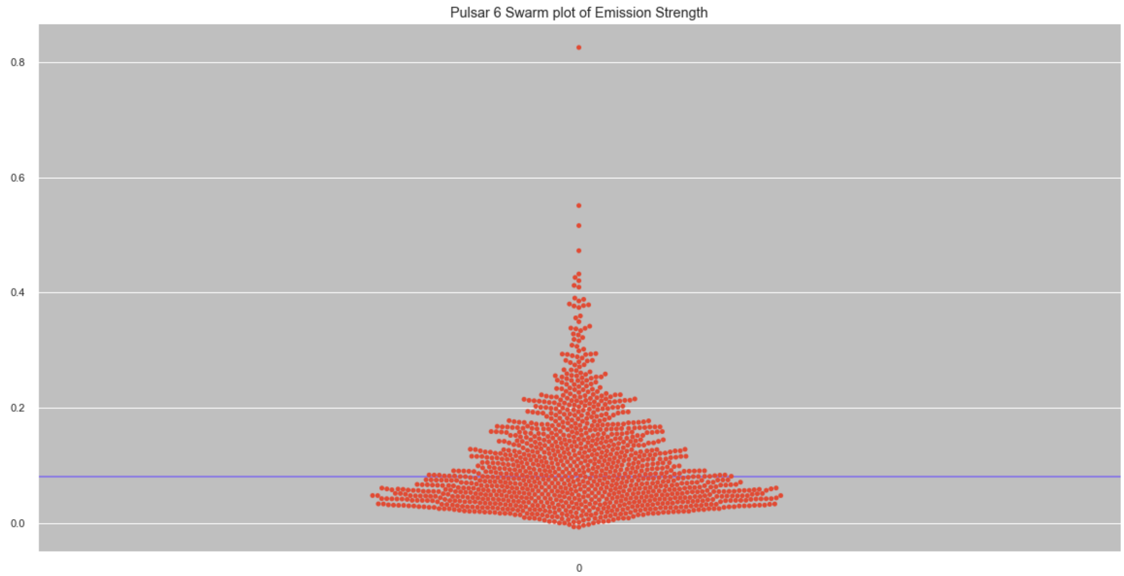
```
c:\Users\oxlay\anaconda3\lib\site-packages\matplotlib\collections.py:1003:
RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```




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

```
609
609
```

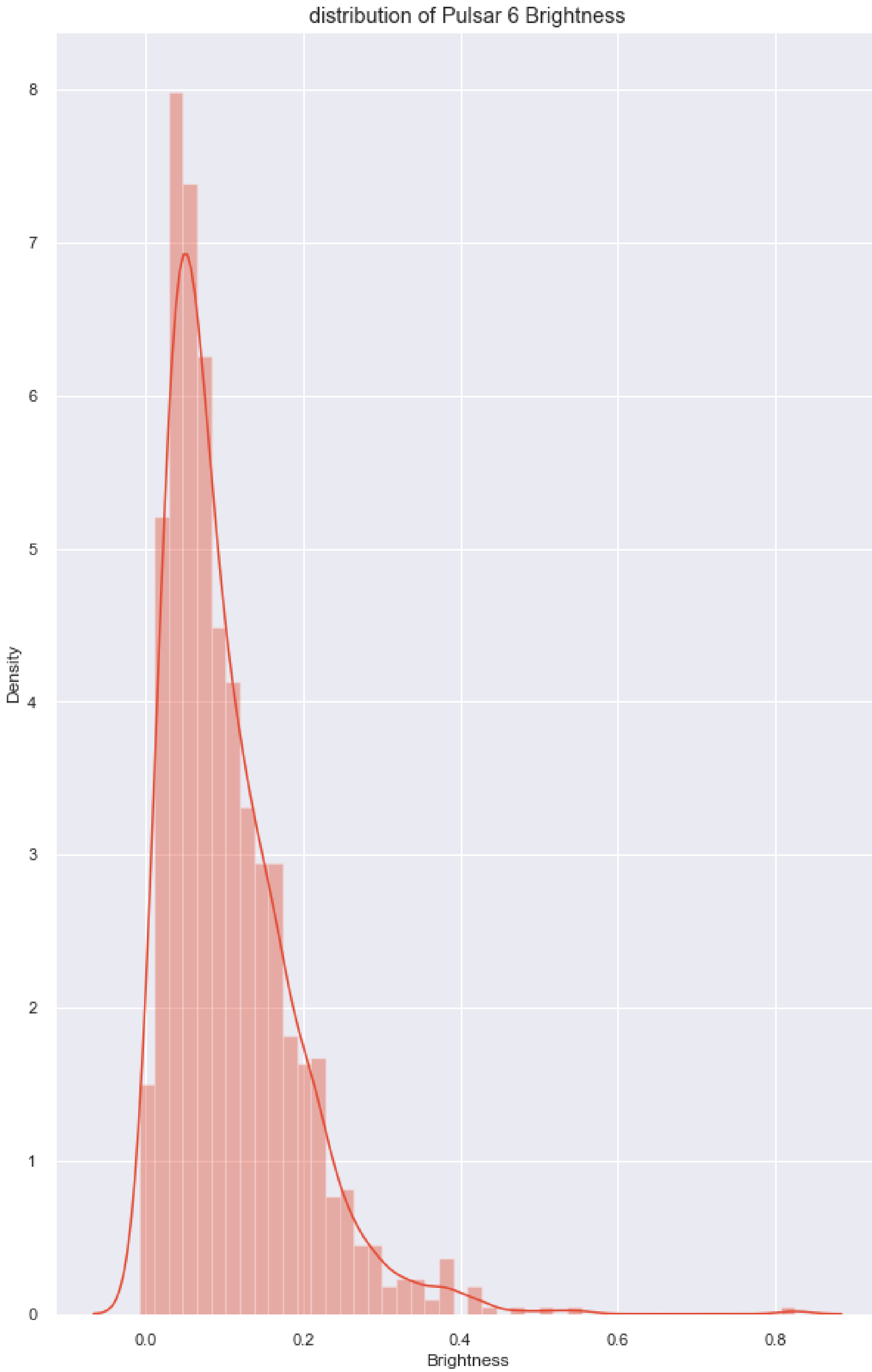
```
[ ]: plt.figure(figsize=(20,10))
     sns.set_style("darkgrid", {"axes.facecolor": ".75"})
     strength = pulsar.Brightness.values
     ax = plt.axhline( y=0.081228, ls='-',c='mediumslateblue')
     ax = sns.swarmplot(data=pulsar["Brightness"], c="blue").set_title('Pulsar 6␣
     ↪Swarm plot of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



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

```
c:\Users\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')

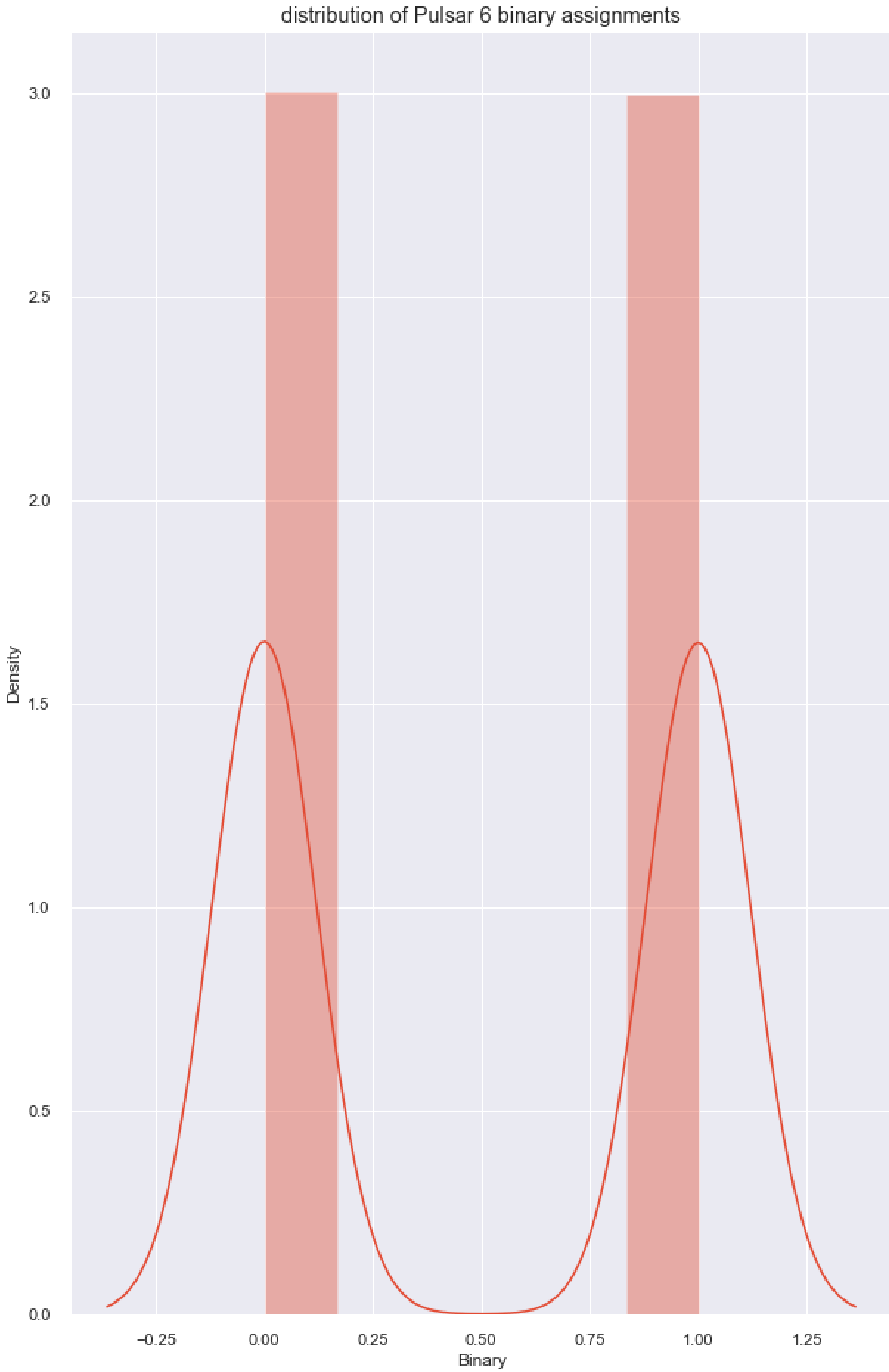


distribution of Pulsar 6 Brightness
Density
Brightness
0
1
2
3
4
5
6
7
8
0.0
0.2
0.4
0.6
0.8

[ ]:
```python
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
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25

3.2 Binary Classification

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

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

[ ]:

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

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

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

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

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

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

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

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

[ ]:
```
model = LogisticRegression()

model.fit(X_train, y_train)
```

[ ]: LogisticRegression()

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

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

```
cm = confusion_matrix(y_test, predictions)

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

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

```
True Positive(TP)  =  117
False Positive(FP) =  3
True Negative(TN)  =  124
False Negative(FN) =  0
```

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

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

```
Accuracy of the model is  0.9877049180327869
```

3.3 Bidirectional LSTM Model

[ ]:
```
brightness_list = list(pulsar['Brightness'])
brightness_list[:10]
```

[ ]:
```
[0.05390386,
 0.05865279,
 0.1102083,
 0.03471609,
 0.05610133,
 0.04616798,
 0.05564797,
 0.06089036,
 0.02438825,
 0.0393704]
```

[ ]:
```
def split_list(blist, steps):
    X, y = list(), list()
    for i in range(len(blist)):
        # find the end of this pattern
        end_ix = i + steps
        # check if we are beyond the sequence
        if end_ix > len(blist)-1:
            break
        # gather input and output parts of the pattern
        list_x, list_y = blist[i:end_ix], blist[end_ix]
        X.append(list_x)
        y.append(list_y)
```

```
    return array(X), array(y)
```

[ ]:
```
X, y = split_list(brightness_list, 100)
X = X.reshape((X.shape[0], X.shape[1], 1))
X[:1]
```

[ ]:
```
array([[[ 0.05390386],
        [ 0.05865279],
        [ 0.1102083 ],
        [ 0.03471609],
        [ 0.05610133],
        [ 0.04616798],
        [ 0.05564797],
        [ 0.06089036],
        [ 0.02438825],
        [ 0.0393704 ],
        [ 0.00914078],
        [ 0.1452735 ],
        [ 0.03995335],
        [-0.00255413],
        [ 0.03569608],
        [ 0.04686878],
        [ 0.08263744],
        [ 0.3494193 ],
        [ 0.05834345],
        [ 0.09026068],
        [ 0.1204294 ],
        [ 0.2097299 ],
        [ 0.0880448 ],
        [ 0.2037356 ],
        [ 0.02409802],
        [ 0.1681109 ],
        [ 0.1047714 ],
        [ 0.1631917 ],
        [ 0.1455741 ],
        [ 0.03896361],
        [ 0.09575639],
        [ 0.08995095],
        [ 0.06254988],
        [ 0.3740181 ],
        [ 0.00821777],
        [ 0.07723381],
        [ 0.02768944],
        [ 0.0722478 ],
        [ 0.03900848],
        [ 0.03498762],
        [ 0.2314543 ],
```

```
[ 0.1414296 ],
[ 0.07437511],
[ 0.07192102],
[ 0.08205932],
[ 0.0642197 ],
[ 0.1158613 ],
[ 0.04618028],
[ 0.1612226 ],
[ 0.05459188],
[ 0.03851383],
[ 0.04747371],
[ 0.2863024 ],
[ 0.131919  ],
[ 0.00942252],
[ 0.06488083],
[ 0.1213254 ],
[ 0.04713579],
[ 0.04365566],
[ 0.1630016 ],
[ 0.00352745],
[ 0.04222952],
[ 0.03658938],
[ 0.04839112],
[ 0.08058076],
[ 0.07470822],
[ 0.09671548],
[ 0.1265122 ],
[ 0.00670705],
[ 0.06265472],
[ 0.1005842 ],
[ 0.02794401],
[ 0.1030977 ],
[ 0.06309345],
[ 0.07762477],
[ 0.0492406 ],
[ 0.1625029 ],
[ 0.1005224 ],
[ 0.03106705],
[ 0.05808663],
[ 0.02931194],
[ 0.04946467],
[ 0.00685568],
[ 0.05183945],
[ 0.2223818 ],
[ 0.1967218 ],
[ 0.14175   ],
[ 0.09903066],
```

```
        [ 0.1372447 ],
        [ 0.03592795],
        [ 0.05079282],
        [ 0.06908489],
        [ 0.1280467 ],
        [ 0.1633251 ],
        [ 0.2823657 ],
        [ 0.02356086],
        [ 0.07807515],
        [ 0.05453903],
        [ 0.05455778],
        [ 0.08110538]]])
```

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

[ ]:
```
model = Sequential()
model.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(100, 1)))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
history = model.fit(X_train, y_train, epochs=50, verbose=1, batch_size=10)
```

```
Epoch 1/50
90/90 [==============================] - 3s 20ms/step - loss: 0.5886 - accuracy:
0.0000e+00
Epoch 2/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3399 - accuracy:
0.0000e+00
Epoch 3/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3378 - accuracy:
0.0000e+00
Epoch 4/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3372 - accuracy:
0.0000e+00
Epoch 5/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3363 - accuracy:
0.0000e+00
Epoch 6/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3362 - accuracy:
0.0000e+00
Epoch 7/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3363 - accuracy:
0.0000e+00
Epoch 8/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3363 - accuracy:
0.0000e+00
```

```
Epoch 9/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3361 - accuracy:
0.0000e+00
Epoch 10/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3366 - accuracy:
0.0000e+00
Epoch 11/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3377 - accuracy:
0.0000e+00
Epoch 12/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3352 - accuracy:
0.0000e+00
Epoch 13/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3368 - accuracy:
0.0000e+00
Epoch 14/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3354 - accuracy:
0.0000e+00
Epoch 15/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3360 - accuracy:
0.0000e+00
Epoch 16/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3357 - accuracy:
0.0000e+00
Epoch 17/50
90/90 [==============================] - 2s 22ms/step - loss: 0.3356 - accuracy:
0.0000e+00
Epoch 18/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3357 - accuracy:
0.0000e+00
Epoch 19/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3357 - accuracy:
0.0000e+00
Epoch 20/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3359 - accuracy:
0.0000e+00
Epoch 21/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3355 - accuracy:
0.0000e+00
Epoch 22/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3359 - accuracy:
0.0000e+00
Epoch 23/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3351 - accuracy:
0.0000e+00
Epoch 24/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3357 - accuracy:
0.0000e+00
```

```
Epoch 25/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3359 - accuracy:
0.0000e+00
Epoch 26/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3367 - accuracy:
0.0000e+00
Epoch 27/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3367 - accuracy:
0.0000e+00
Epoch 28/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3357 - accuracy:
0.0000e+00
Epoch 29/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3355 - accuracy:
0.0000e+00
Epoch 30/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3354 - accuracy:
0.0000e+00
Epoch 31/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3357 - accuracy:
0.0000e+00
Epoch 32/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3356 - accuracy:
0.0000e+00
Epoch 33/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3353 - accuracy:
0.0000e+00
Epoch 34/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3355 - accuracy:
0.0000e+00
Epoch 35/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3353 - accuracy:
0.0000e+00
Epoch 36/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3354 - accuracy:
0.0000e+00
Epoch 37/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3355 - accuracy:
0.0000e+00
Epoch 38/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3358 - accuracy:
0.0000e+00
Epoch 39/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3356 - accuracy:
0.0000e+00
Epoch 40/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3354 - accuracy:
0.0000e+00
```

```
Epoch 41/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3351 - accuracy:
0.0000e+00
Epoch 42/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3354 - accuracy:
0.0000e+00
Epoch 43/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3355 - accuracy:
0.0000e+00
Epoch 44/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3352 - accuracy:
0.0000e+00
Epoch 45/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3349 - accuracy:
0.0000e+00
Epoch 46/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3351 - accuracy:
0.0000e+00
Epoch 47/50
90/90 [==============================] - 2s 21ms/step - loss: 0.3354 - accuracy:
0.0000e+00
Epoch 48/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3350 - accuracy:
0.0000e+00
Epoch 49/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3352 - accuracy:
0.0000e+00
Epoch 50/50
90/90 [==============================] - 2s 20ms/step - loss: 0.3350 - accuracy:
0.0000e+00
```

[ ]:
```
y_pred = model.predict(X_test, verbose=0)
y_pred[:10]
```

[ ]:
```
array([[0.11022874],
       [0.11202019],
       [0.11282037],
       [0.11894967],
       [0.11175107],
       [0.11313197],
       [0.11536959],
       [0.1112138 ],
       [0.10938548],
       [0.11274448]], dtype=float32)
```

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

```
7/7 [==============================] - 0s 7ms/step - loss: 0.3449 - accuracy:
```

```
0.0000e+00
```

[ ]: [0.3448549807071686, 0.0]

3.4 ML Evaluation.

3.4.1 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.

3.4.2 Bidirectional LSTM

This model is very error prone as the loss value is consistently at 60& or higher at every epoch during training and at exactly 63.07% in evaluation with a 0% accuracy this indicates that there is either a great error in the formation of the model, data used or trend being obtained. Alternatively it could indicate that there is no trend there to predict. Likely this indicates that the model is not valuable for any meaningful analysis.

4 Preliminary runs test

4.0.1 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.

5 FUNCTION CODE FOR RUNS TEST

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

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

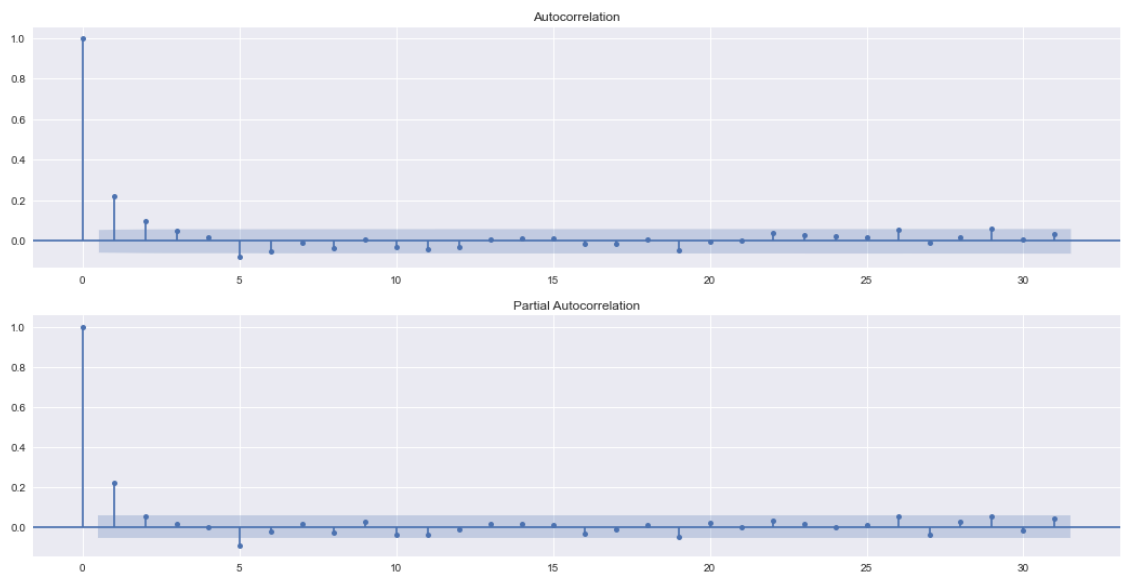
```
0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0]
```

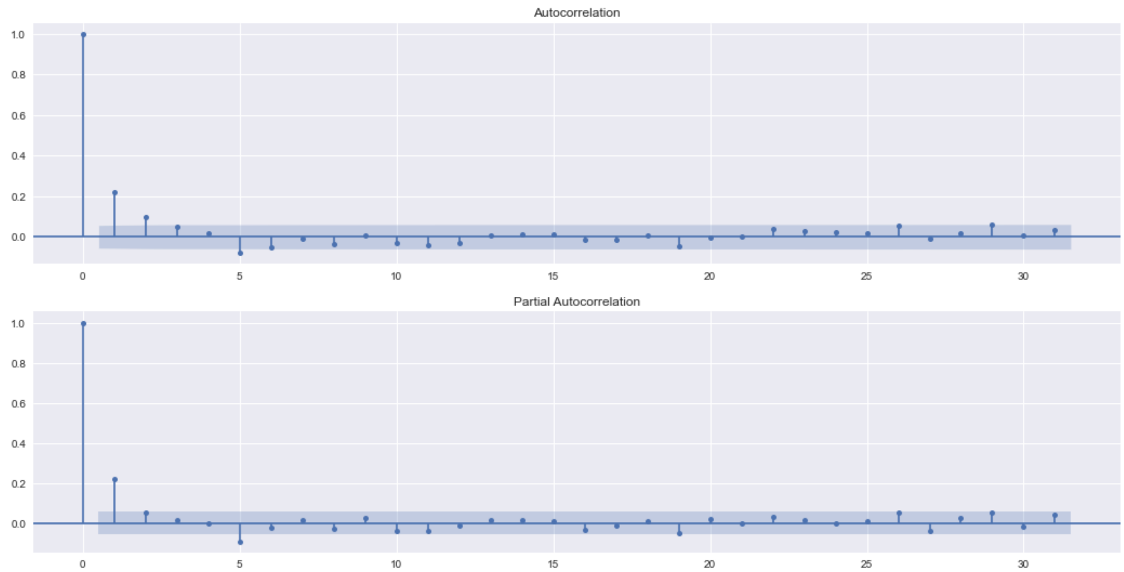
6 Below we begin autocorrelation and autocovariance analysis

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.        ,  0.22161581,  0.09940415,  0.04669096,  0.01876941,
       -0.07818365, -0.0539178 , -0.01220963, -0.03566829,  0.00520742,
       -0.03014362])
```

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


acfpulsar
```

[ ]:

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

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

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

[1219 rows x 11 columns]
```

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

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

6.0.1 Getting every 5th as per the auto correlation

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

6.0.3 Further Random testing to move into extensive testing

Getting every 5th as per the auto correlation

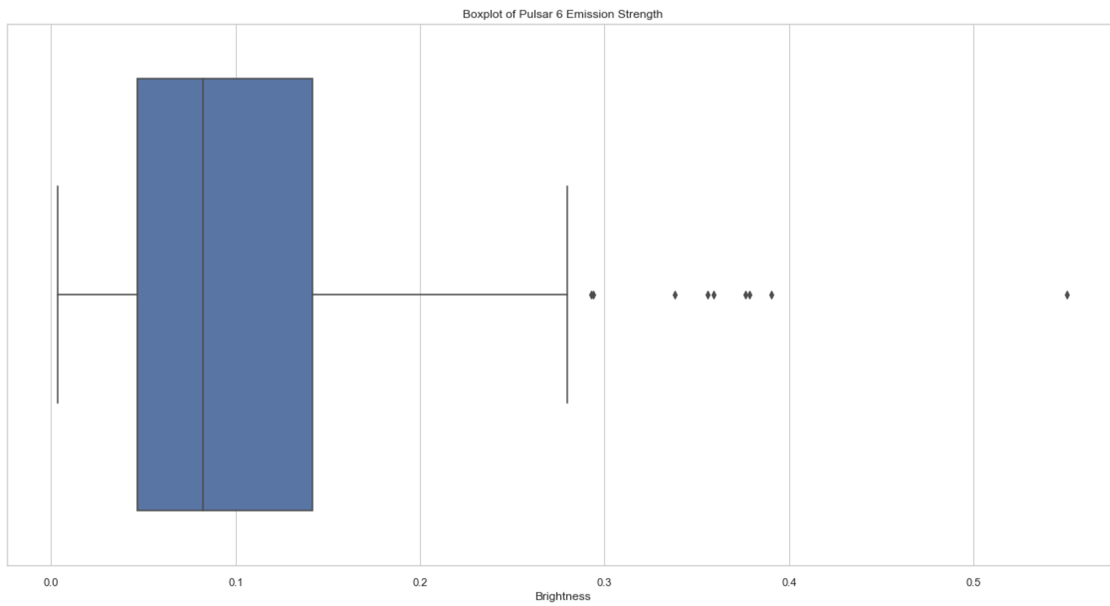
[ ]:
```
held5ths = pulsar[pulsar.index % 5 == 0]
held5ths
```

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

[244 rows x 4 columns]
```

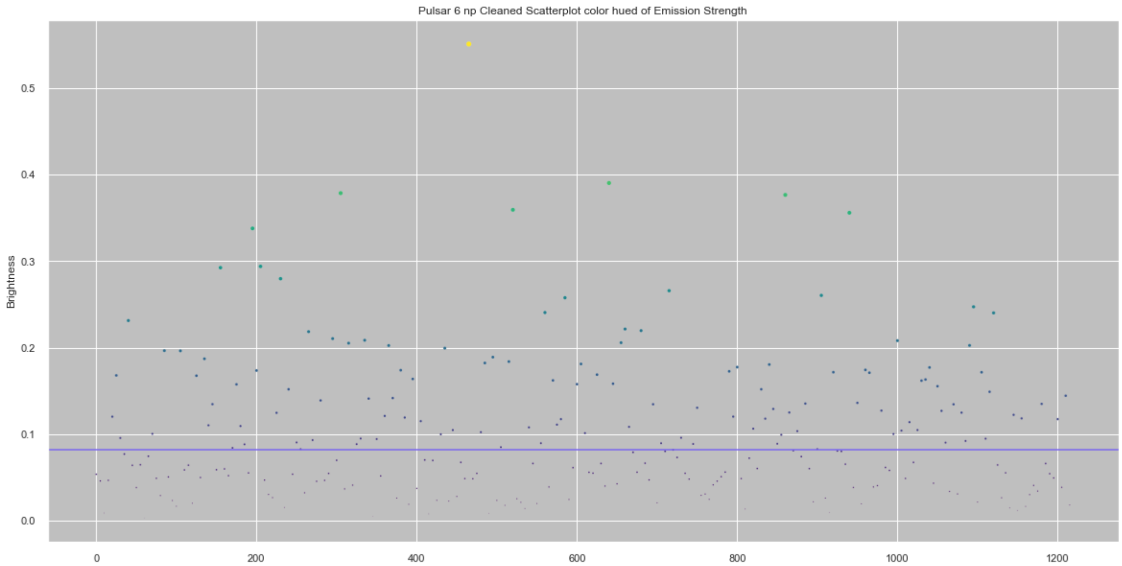
[ ]:
```
medianheld5ths = held5ths["Brightness"].median()
medianheld5ths
```

[ ]: 0.08254402

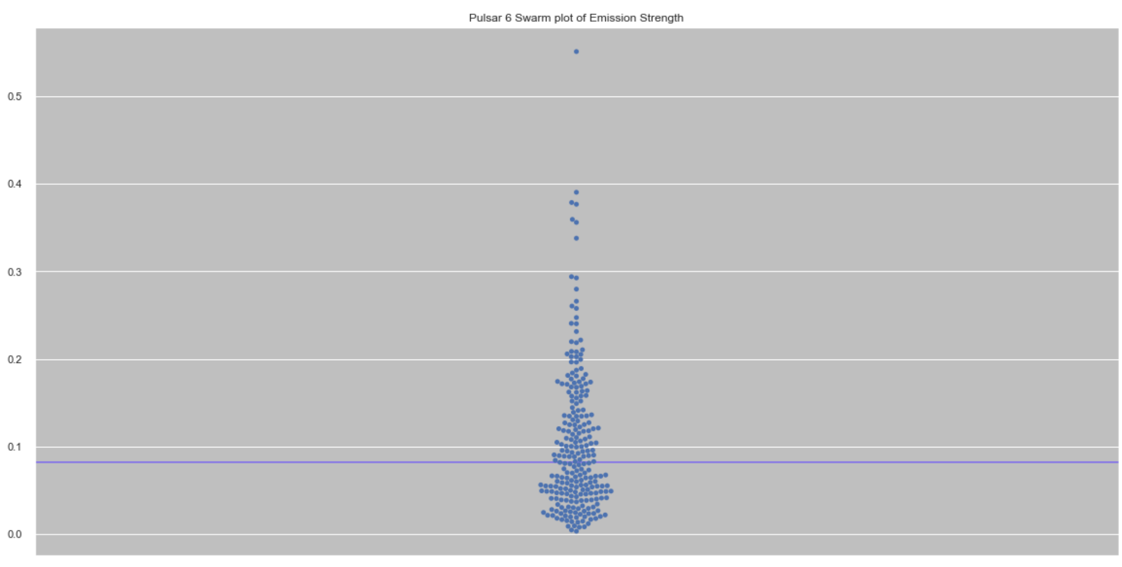
[ ]:
```
plt.figure(figsize=(20,10))
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=held5ths["Brightness"]).set_title("Boxplot of Pulsar 6␣
↪Emission Strength")
```



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



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



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

```
122
122
```

Randomness testing

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

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

```
[ ]: 0       0
     1       0
     2       1
     3       0
     4       0
            ..
     1214    0
     1215    0
     1216    1
     1217    0
     1218    0
     Name: Binary, Length: 1219, dtype: int32
```