pulsar3

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/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
```

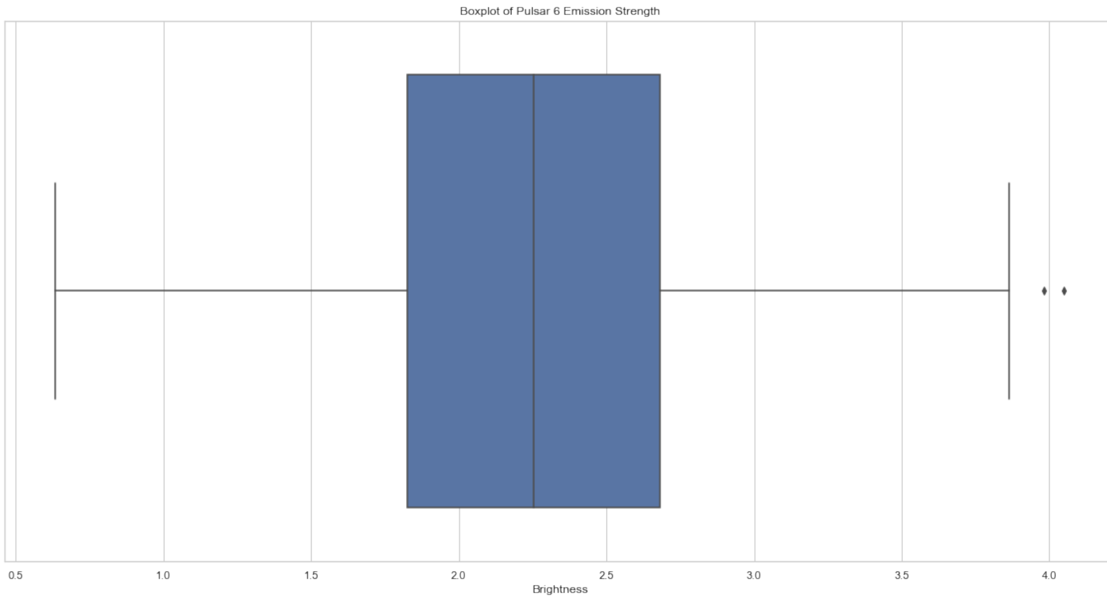
[ ]:
```
nullBoolBrightness = pd.isnull(pulsar["Brightness"])

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
```

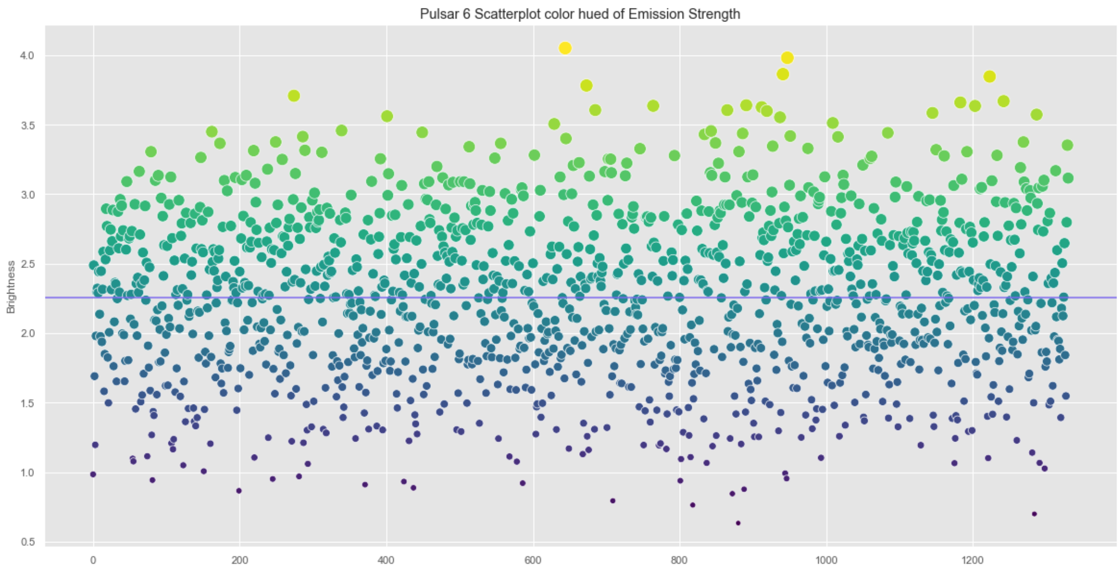
[ ]:

	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')
```

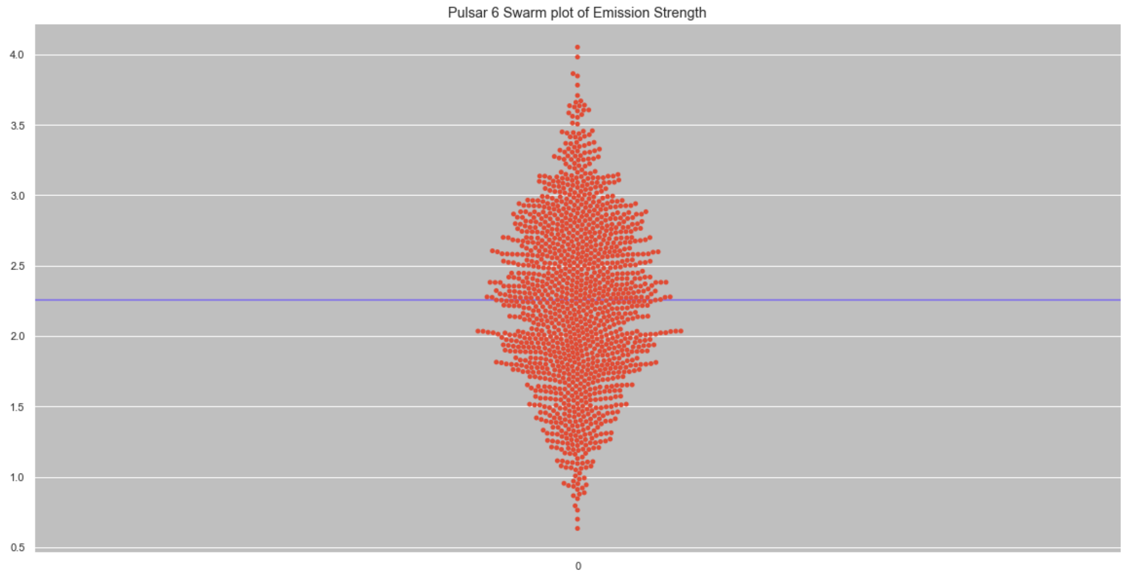
```
ax = sns.scatterplot(data=pulsar["Brightness"], s= strength*50, c=strength,␣
↪cmap="viridis", marker="o").set_title('Pulsar 6 Scatterplot color hued of␣
↪Emission Strength')
ax= plt.axhline( y=2.255182, ls='-',c='mediumslateblue')
```




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

665
665

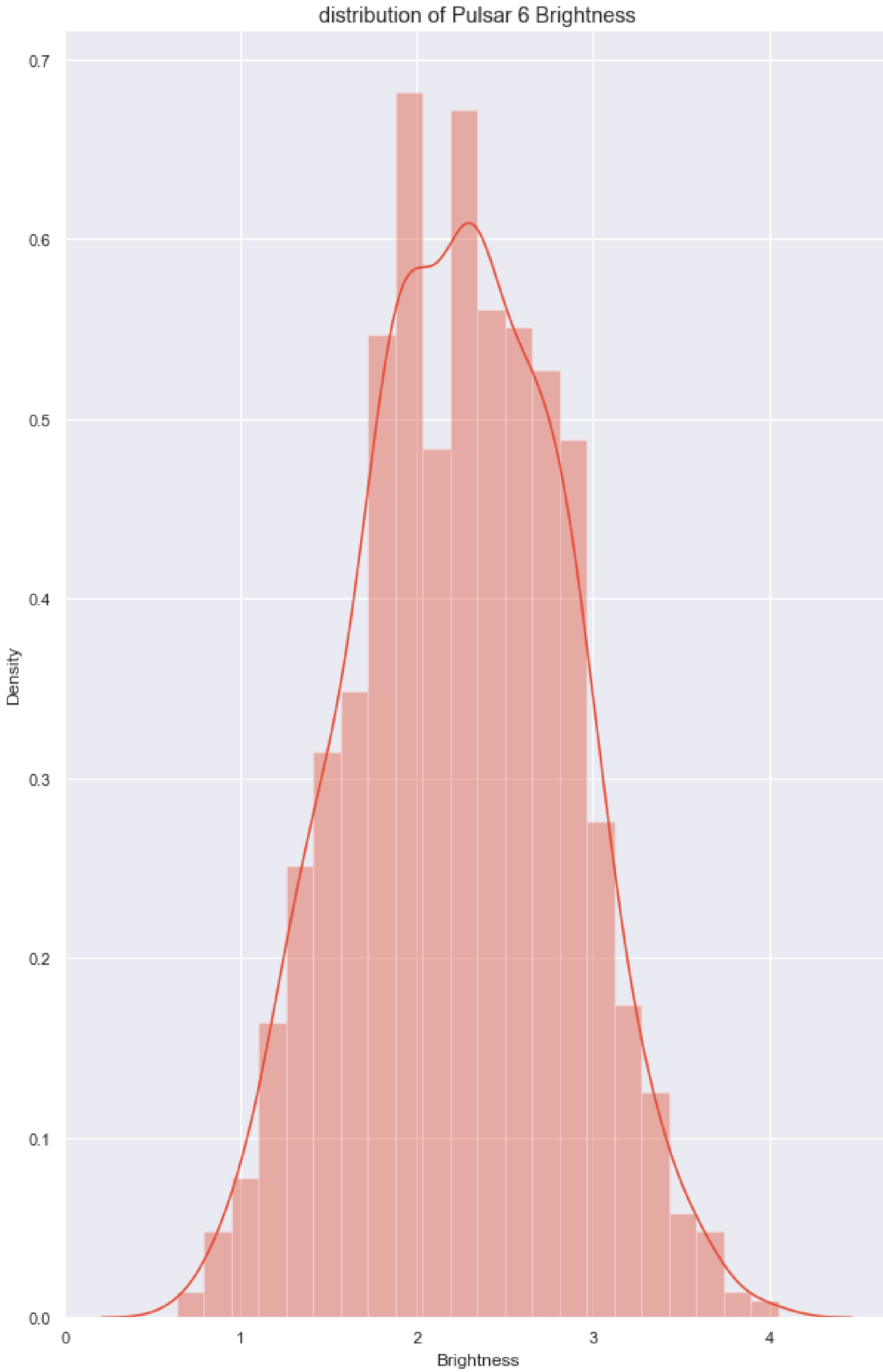
```
[ ]: plt.figure(figsize=(20,10))
     sns.set_style("darkgrid", {"axes.facecolor": ".75"})
     strength = pulsar.Brightness.values
     ax = plt.axhline( y=2.255182, ls='-',c='mediumslateblue')
     ax = sns.swarmplot(data=pulsar["Brightness"], c="blue").set_title('Pulsar 6␣
     ↪Swarm plot of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



[ ]:
```python
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)
```

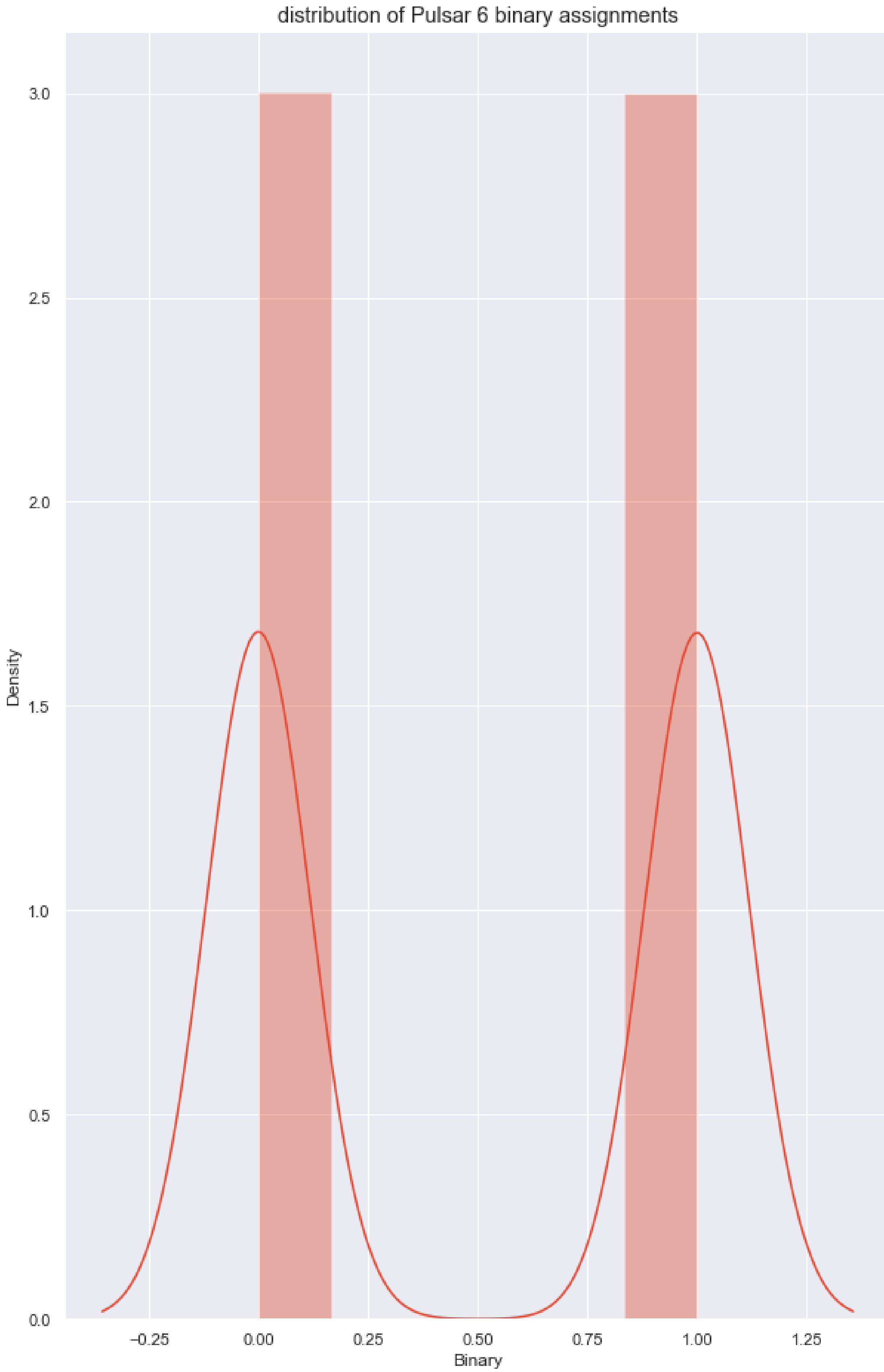
[ ]:
```
Text(0.5, 1.0, 'distribution of Pulsar 6 Brightness')
```


distribution of Pulsar 6 Brightness
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
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\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    0.984043     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
```

```python
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)  =  132
False Positive(FP) =  0
True Negative(TN)  =  122
False Negative(FN) =  13
```

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

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

```
Accuracy of the model is  0.951310861423221
```

3.3 Bidirectional LSTM Model

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

[ ]:
```
[0.9840433,
 2.487928,
 1.690295,
 1.196142,
 1.979783,
 2.297645,
 2.322135,
 2.289047,
 2.442574,
 2.136332]
```

[ ]:
```python
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([[[0.9840433],
        [2.487928 ],
        [1.690295 ],
        [1.196142 ],
        [1.979783 ],
        [2.297645 ],
        [2.322135 ],
        [2.289047 ],
        [2.442574 ],
        [2.136332 ],
        [1.97679  ],
        [2.445764 ],
        [1.937017 ],
        [2.315184 ],
        [2.584888 ],
        [1.581452 ],
        [1.849656 ],
        [2.529834 ],
        [2.894401 ],
        [2.769474 ],
        [1.82449  ],
        [1.498133 ],
        [2.005834 ],
        [2.594836 ],
        [2.745045 ],
        [2.312928 ],
        [2.661311 ],
        [2.885388 ],
        [1.761946 ],
        [2.081099 ],
        [2.36358  ],
        [2.35345  ],
        [1.652502 ],
        [2.248286 ],
        [2.874749 ],
        [2.610421 ],
        [2.294724 ],
        [2.961658 ],
        [2.696347 ],
        [2.917047 ],
        [1.996423 ],
```

```
[2.739376 ],
[1.986204 ],
[1.653353 ],
[1.800485 ],
[2.600838 ],
[3.090313 ],
[2.271085 ],
[1.805165 ],
[2.681646 ],
[2.696566 ],
[2.101715 ],
[2.278763 ],
[2.734046 ],
[1.096633 ],
[1.077475 ],
[2.062858 ],
[2.92687  ],
[1.454358 ],
[2.356589 ],
[1.98154  ],
[2.399687 ],
[2.292884 ],
[3.16329  ],
[2.709231 ],
[1.506129 ],
[2.353669 ],
[1.554678 ],
[2.579631 ],
[1.70781  ],
[2.381017 ],
[2.914784 ],
[2.236938 ],
[2.507387 ],
[1.114129 ],
[1.90936  ],
[1.752472 ],
[2.473441 ],
[1.586753 ],
[3.304978 ],
[1.267595 ],
[0.9429132],
[1.435816 ],
[1.407151 ],
[2.094246 ],
[3.098825 ],
[2.088198 ],
[1.557415 ],
```

```
        [2.167673 ],
        [3.134241 ],
        [1.791176 ],
        [2.026329 ],
        [1.796364 ],
        [2.971239 ],
        [2.195826 ],
        [2.841772 ],
        [2.411585 ],
        [2.63257  ],
        [1.859452 ],
        [1.621589 ]]])
```

[ ]:
```
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
99/99 [==============================] - 3s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 2/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 3/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 4/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 5/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 6/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 7/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 8/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
```

```
Epoch 9/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 10/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 11/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 12/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 13/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 14/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 15/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 16/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 17/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 18/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 19/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 20/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 21/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 22/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 23/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 24/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
```

```
Epoch 25/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 26/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 27/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 28/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 29/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 30/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 31/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 32/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 33/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 34/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 35/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 36/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 37/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 38/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 39/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 40/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
```

```
Epoch 41/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 42/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 43/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 44/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 45/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 46/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 47/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 48/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 49/50
99/99 [==============================] - 2s 20ms/step - loss: nan - accuracy:
0.0000e+00
Epoch 50/50
99/99 [==============================] - 2s 21ms/step - loss: nan - accuracy:
0.0000e+00
```

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

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

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

```
8/8 [==============================] - 0s 7ms/step - loss: nan - accuracy:
```

```
0.0000e+00
```

[ ]:
```
[nan, 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, 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, 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, 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,
```

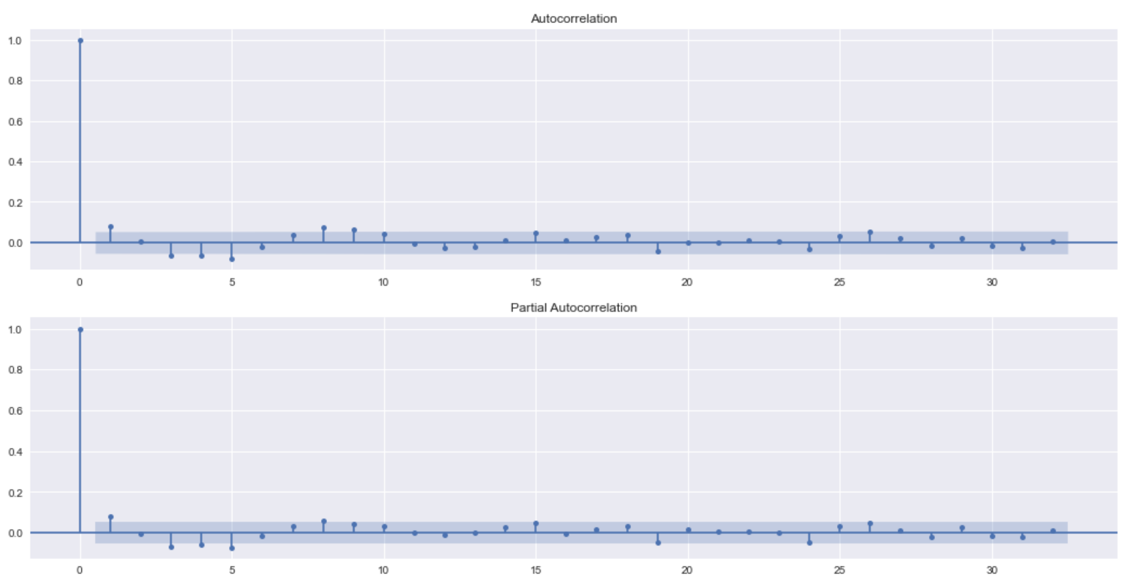
```
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1,
0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,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

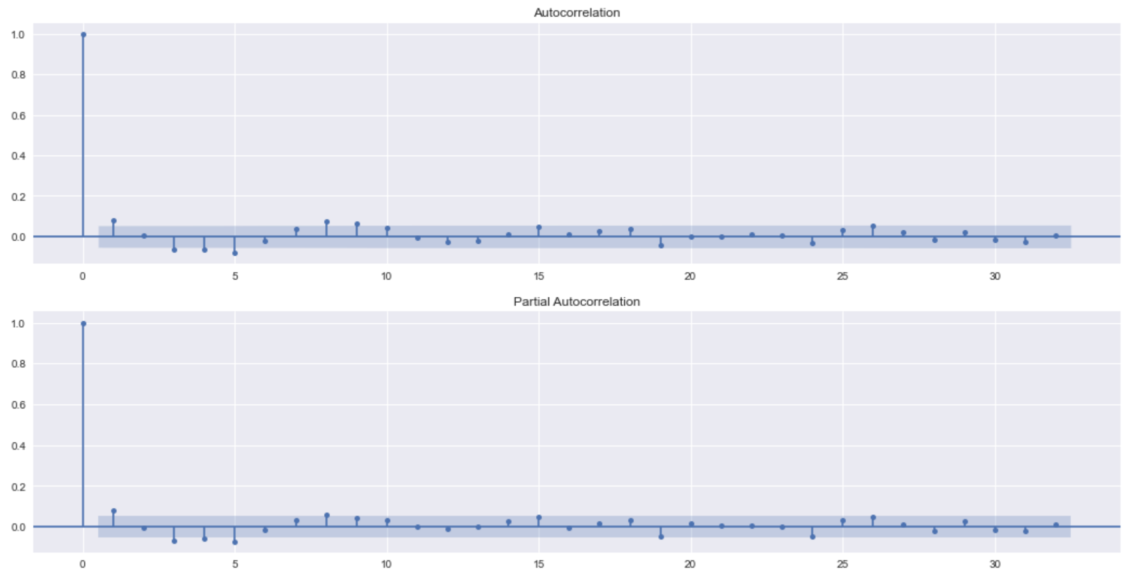
6 Below we begin autocorrelation and autocovariance analysis

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")
```

[ ]:





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

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

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

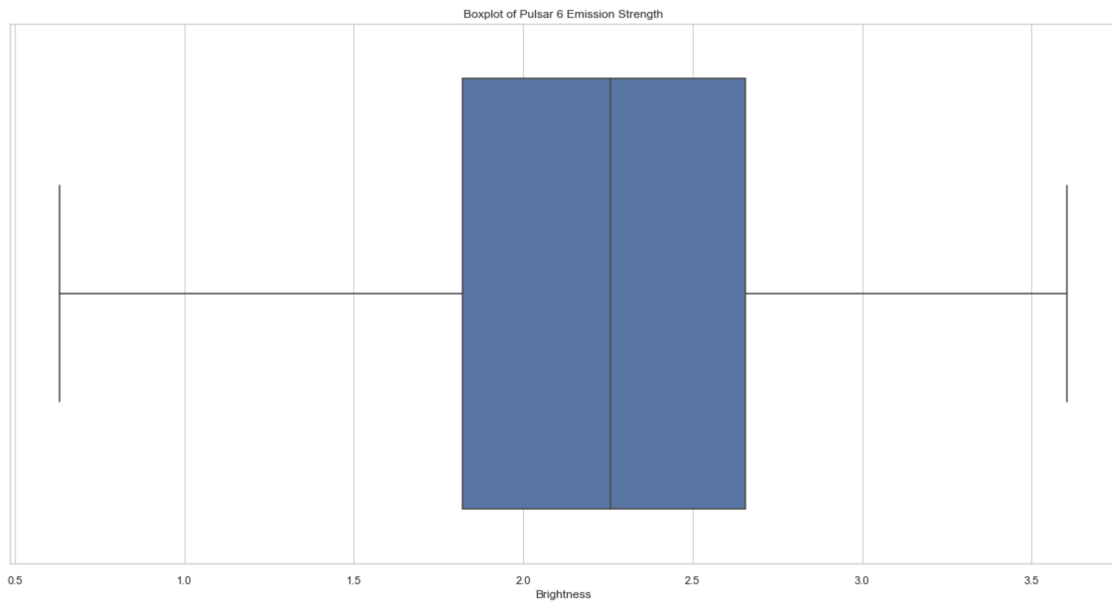
[ ]:
```
held5ths = pulsar[pulsar.index % 5 == 0]
held5ths
```

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

[267 rows x 4 columns]
```

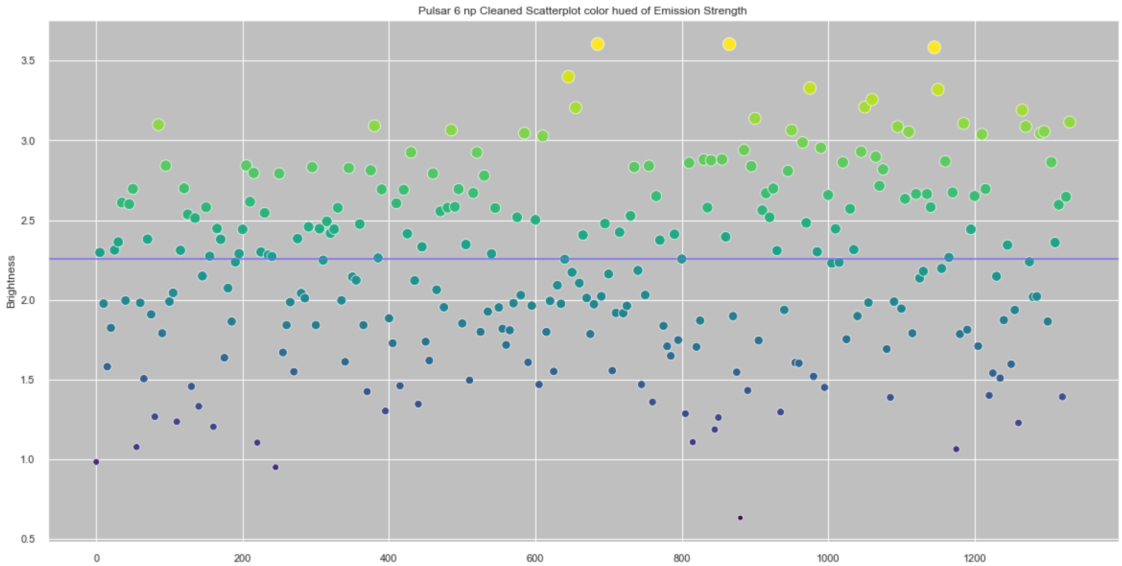
```
[ ]: medianheld5ths = held5ths["Brightness"].median()
     medianheld5ths
```

```
[ ]: 2.256816
```

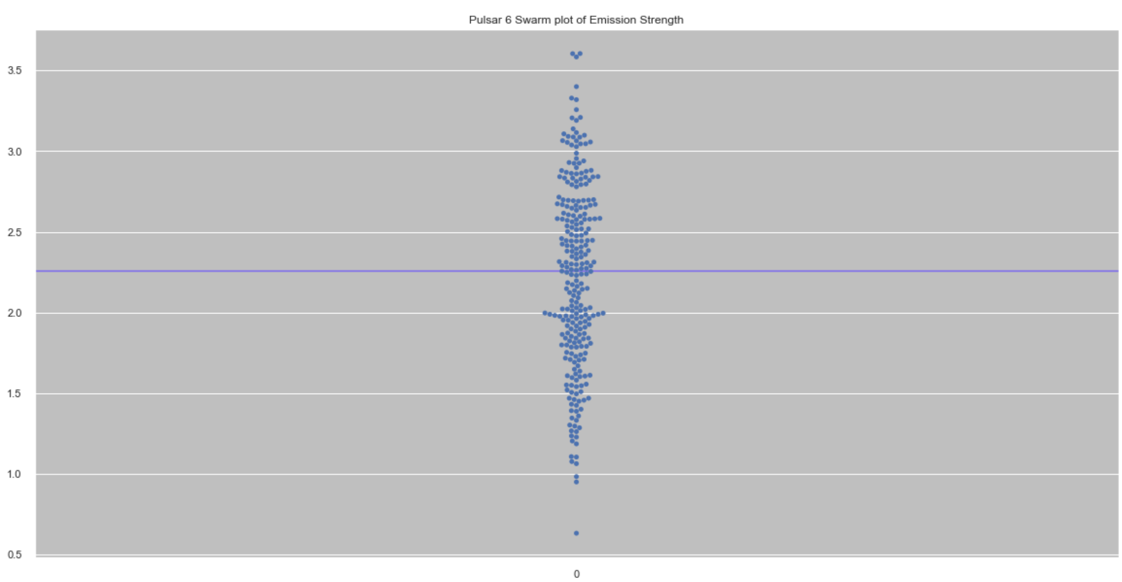
```
[ ]: plt.figure(figsize=(20,10))
     sns.set_theme(style="whitegrid")
     ax = sns.boxplot(x=held5ths["Brightness"]).set_title("Boxplot of Pulsar 6␣
      ↪Emission Strength")
```




```
[ ]: 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=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
```