

PART-2: Training a model to classify an ECG with AF

-By S Ashwin Kumar

Null Values Handling

The data is checked for any Null values and if found are replaced by the mean of the respective feature/column as shown below.

Null value report:

FEATURE	NULL VALUE COUNT
bpm	23
ibi	23
sddn	23
sdsd	106
rmssd	106
pnn20	106
pnn50	106
hr_mad	23
sd1	106
sd2	106
s	106
sd1/sd2	161
breathingrate	115

The null values are replaced with mean using the below code:

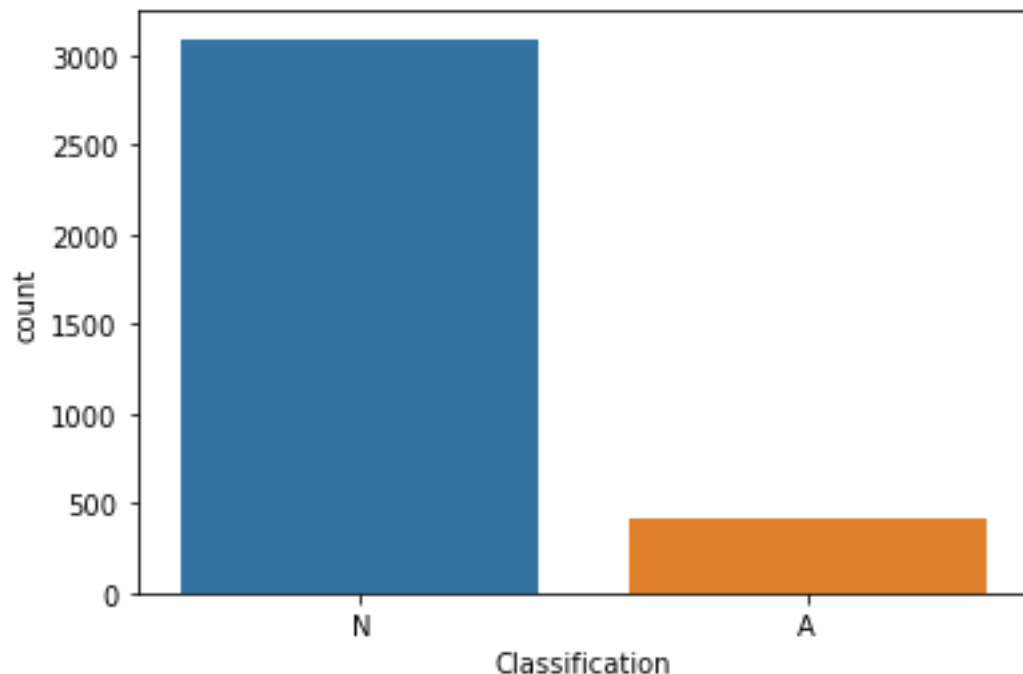
```
# Replacing null values with mean
for i in xx_noise.columns[:-1]:
    xx_noise[i].fillna(
        value = pd.to_numeric(
            xx_noise[i]
        ).mean()
    ,inplace=True
    )
```

Data Analysing & Outlier Detection

The data is analysed to check the skewness and detect outliers.

Count Plot for the Output/Predicted variable with two possible outputs that are:

- Normal (N)
- AF episode (A)



The count plot shows that the data having normal reading is a lot higher than the data dealing with AF episode.

The skewness of data is checked as below:

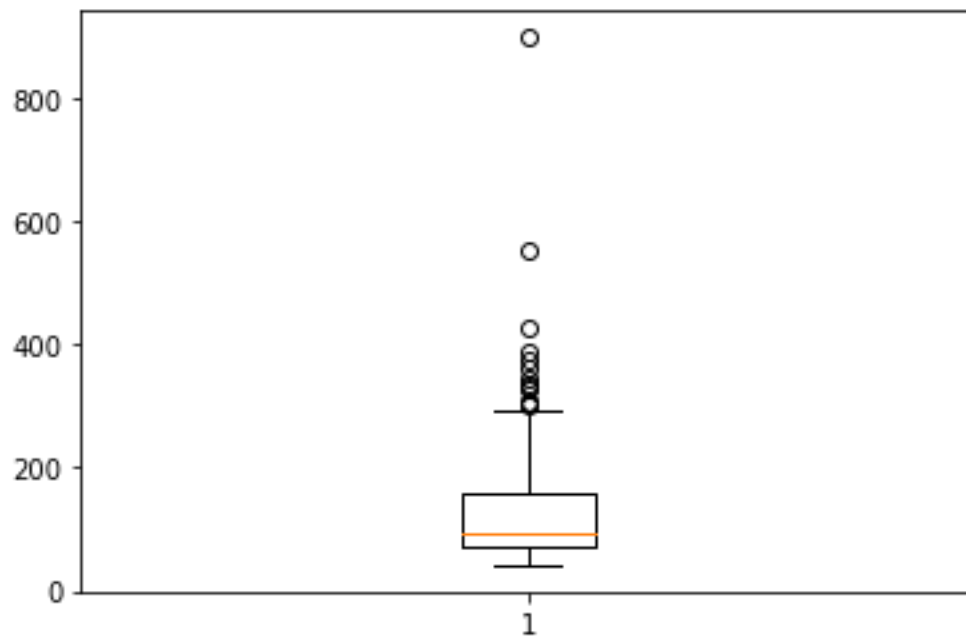
```
1 # Checking Skewness of features
2
3 for column in xx_noise.columns[:-1]:
4     print(column, 'skewness:', xx_noise[column].skew())
```

```
bpm skewness: 1.7225403355998257
ibi skewness: 0.18061994491252278
sdnn skewness: 0.14470856318984826
sdsd skewness: 0.5673517267619748
rmssd skewness: 0.4729273752227066
pnn20 skewness: -0.48056216436576576
pnn50 skewness: 0.006890443533169791
hr_mad skewness: 0.9123081957165963
sd1 skewness: 0.5134504333916752
sd2 skewness: 0.6805479653622352
s skewness: 0.8826475613133172
sd1/sd2 skewness: 15.283806168391749
breathingrate skewness: 3.6419024221751273
```

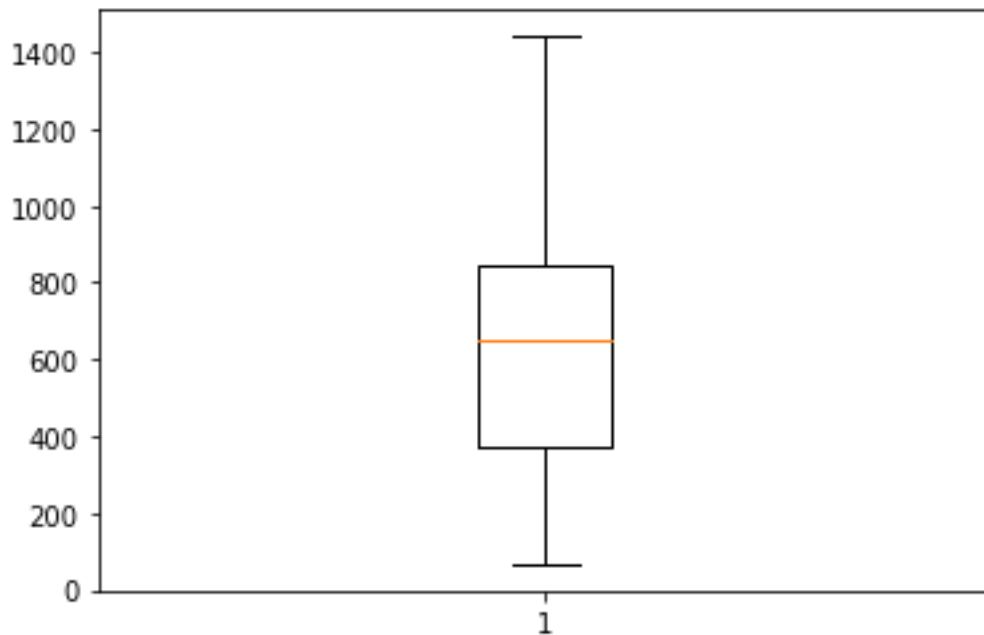
This skewness is a lot higher for 'sd1/sd2' feature.

Checking outliers using the box plot graph as shown below.

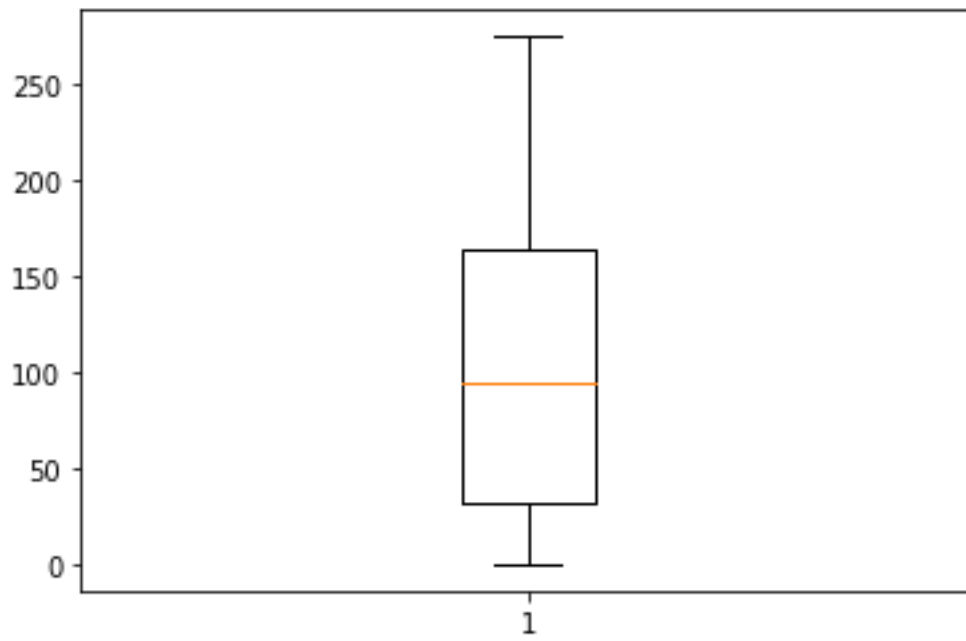
BOX PLOT for bpm



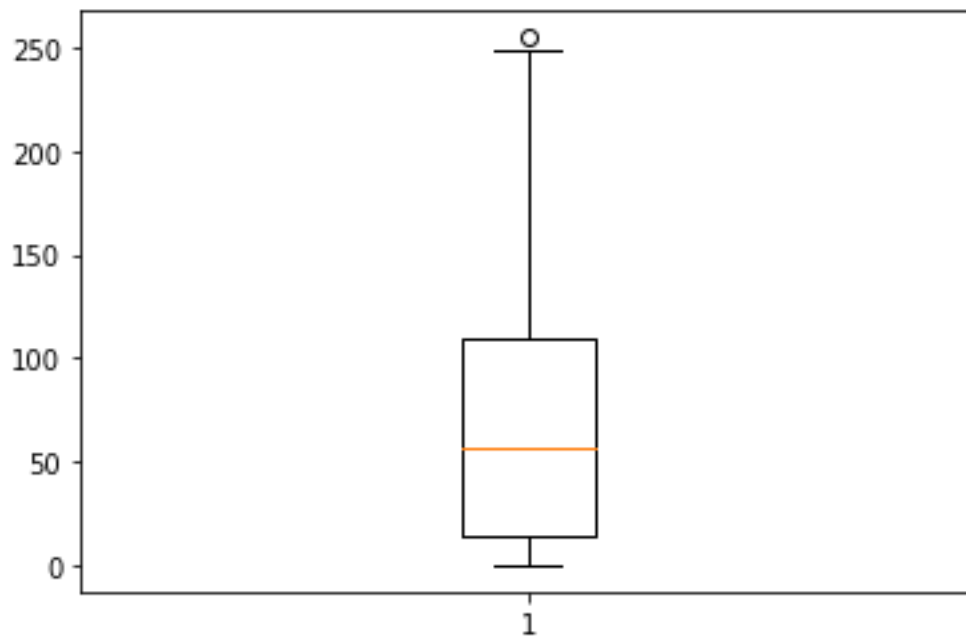
BOX PLOT for ibi



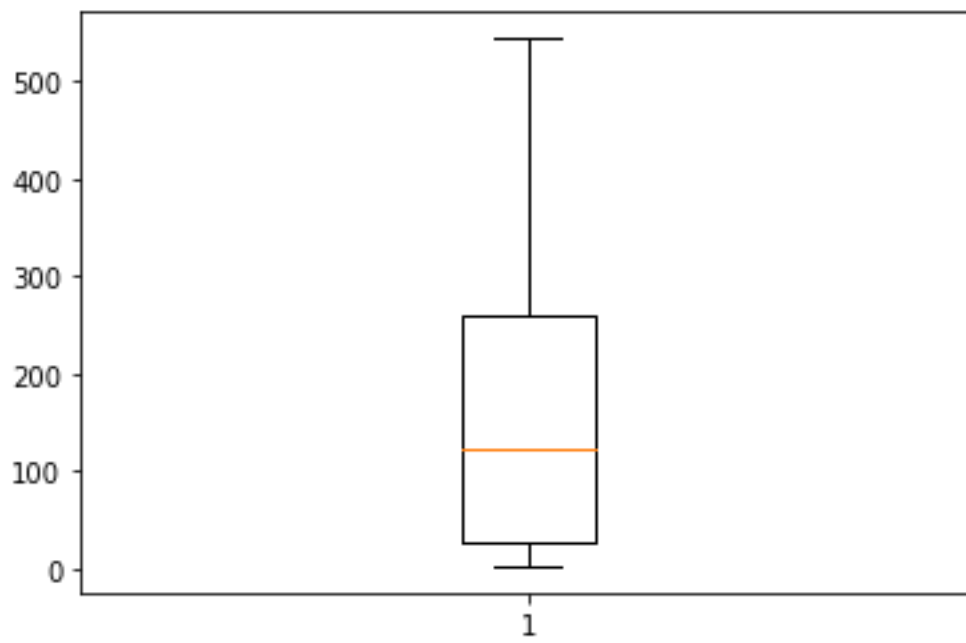
BOX PLOT for sdn



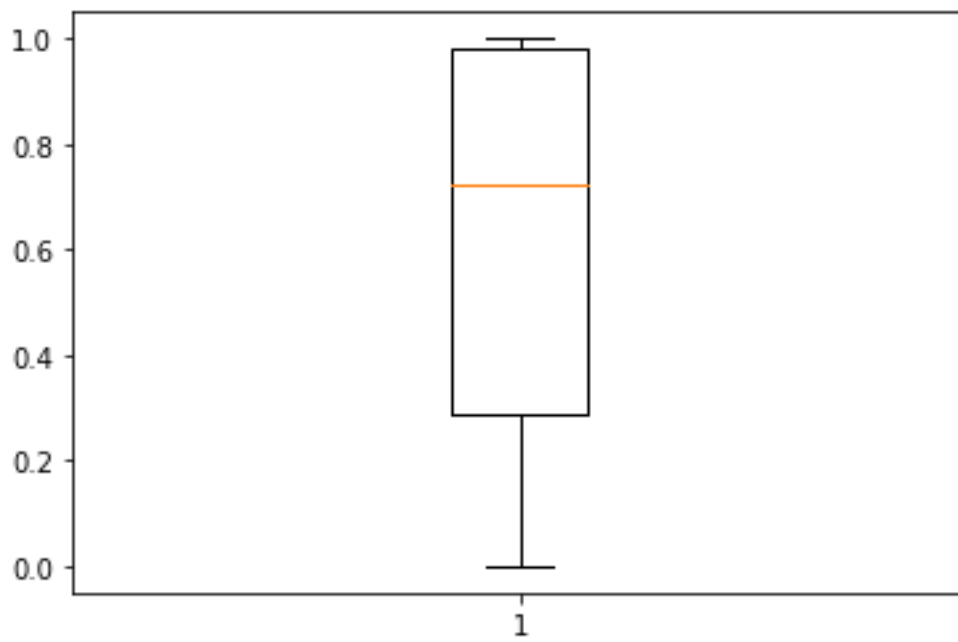
BOX PLOT for sdsd



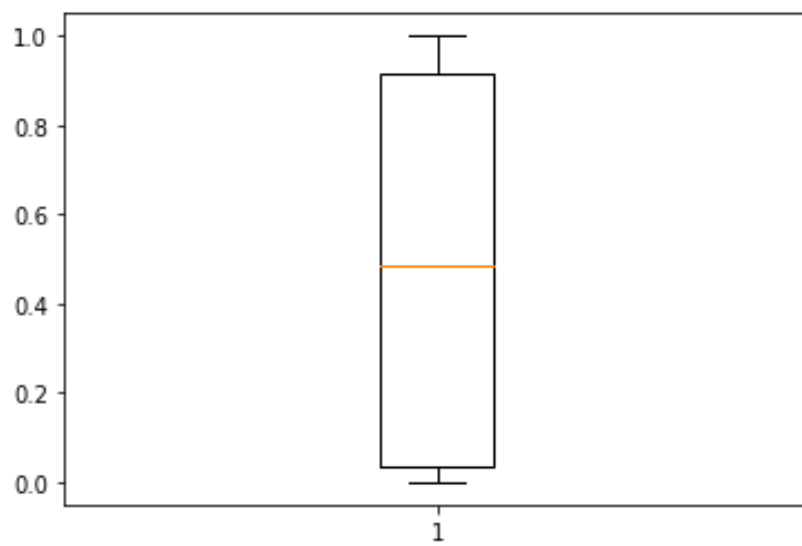
BOX PLOT for rmssd



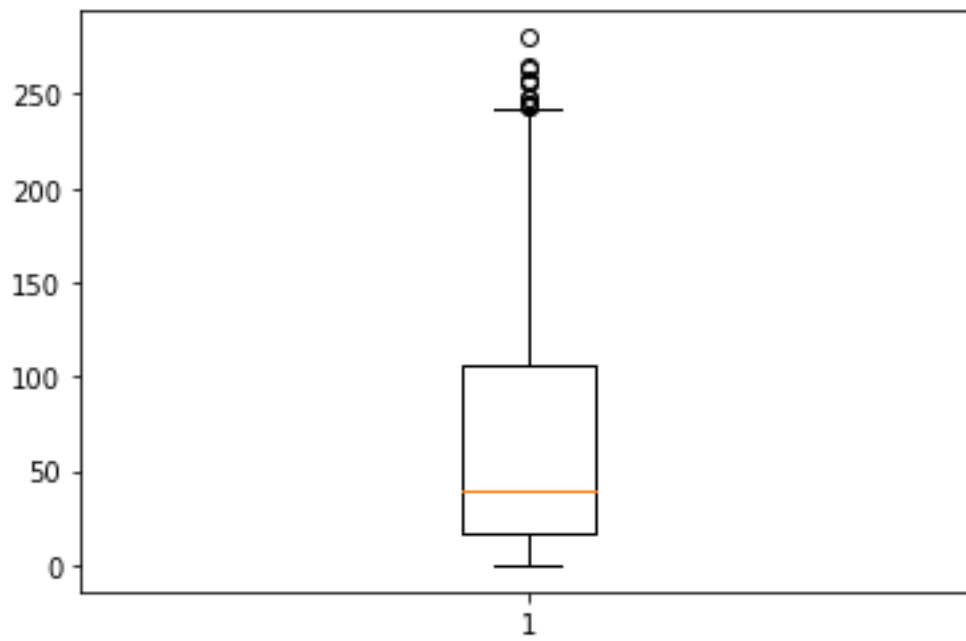
BOX PLOT for pnn20



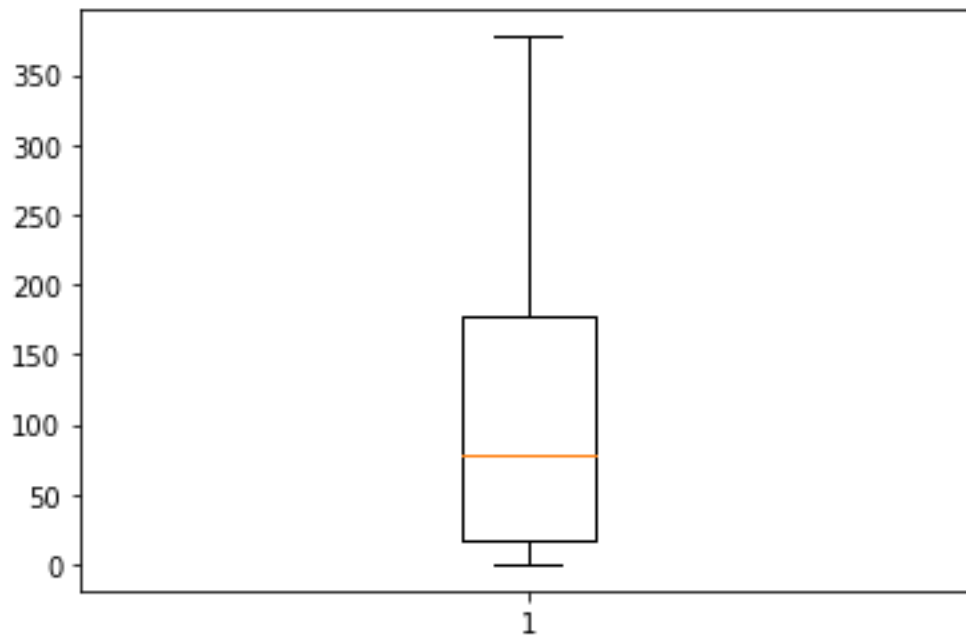
BOX PLOT for pnn50



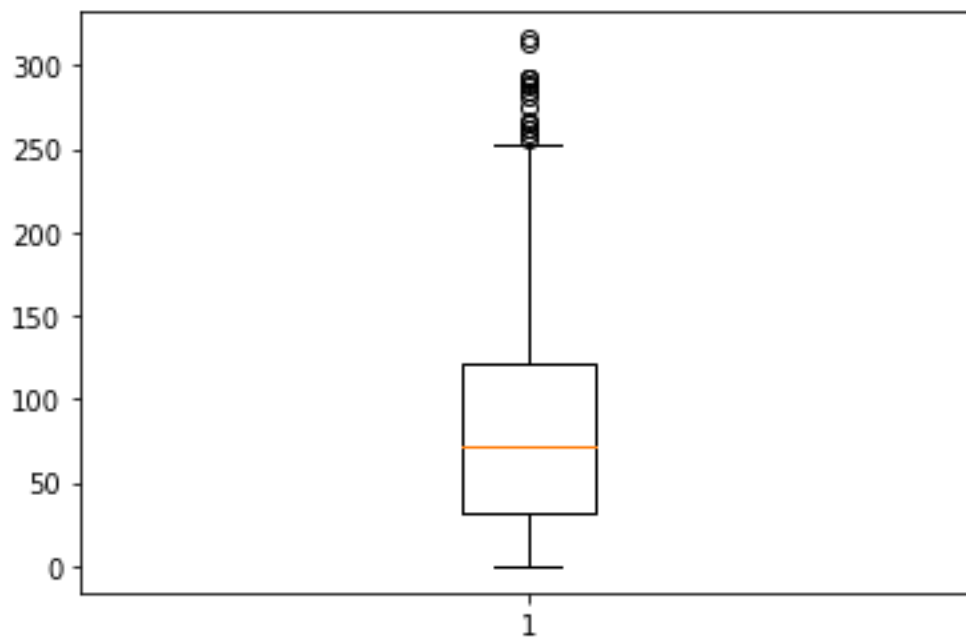
BOX PLOT for hr_mad



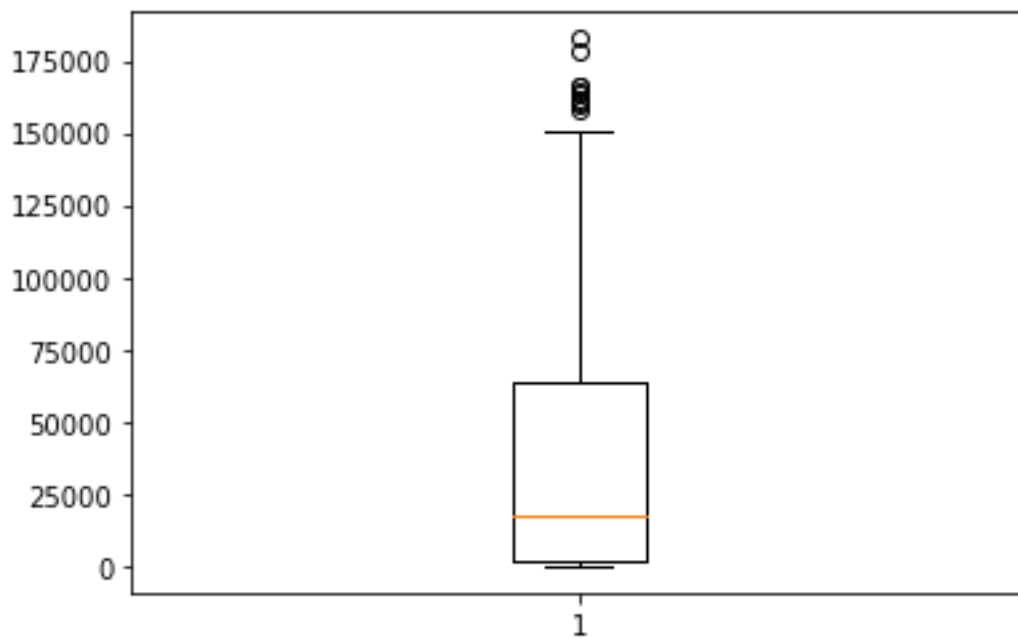
BOX PLOT for sd1



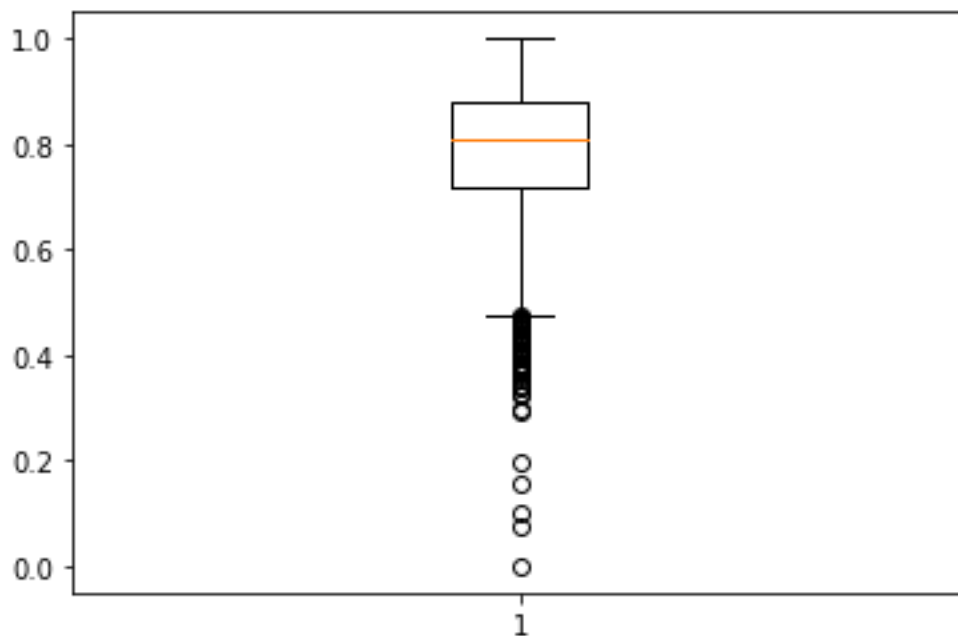
BOX PLOT for sd2



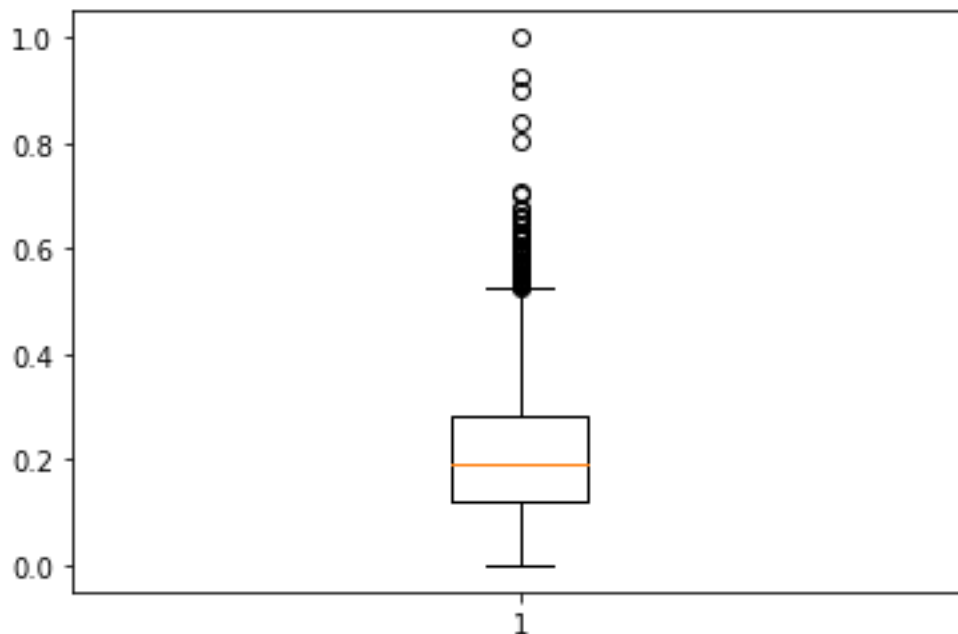
BOX PLOT for s



BOX PLOT for sd1/sd2



BOX PLOT for breathingrate



Data Transformation

As observed from the analysis the data is skewed and has outliers. We can perform log transformation to reduce the skewness and data normalisation using L1 regularization (Lasso regression) with a penalty function to transform the data in a range of 0 to 1.

```
from sklearn.preprocessing import Normalizer
Data_normalizer = Normalizer(norm='l1').fit(xx_noise.iloc[:, :13])
Data_normalized = Data_normalizer.transform(xx_noise.iloc[:, :13])
```


The above piece of code is used to normalize the data using L1 regularization technique. Now that the data is transformed we have to split it into test and train variables using below code.

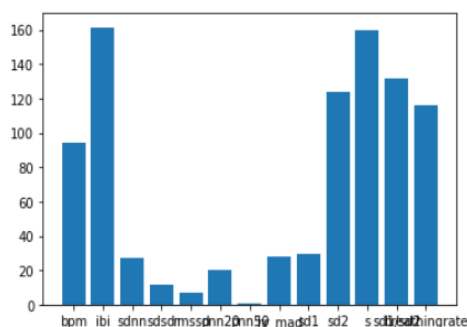
```
# Data splitting into train and test with 80:20 split
x_train,x_test,y_train,y_test=train_test_split(Data_normalized,y,test_size=0.2)
```

Now we can perform feature selection method using **ANOVA (Analysis of Variance)** to check the relation between features and understand it. Then with the understanding of the data ANOVA can transform the data to give more accuracy as shown below.

```
# configure to select all features
fs = SelectKBest(score_func=f_classif, k='all')
# learn relationship from training data
fs.fit(x_train, y_train)
# transform train input data
X_train_fs = fs.transform(x_train)
# transform test input data
X_test_fs = fs.transform(x_test)
```

```
1
2 # what are scores for the features
3 for i in range(len(fs.scores_)):
4     print('Feature %s: %f' % (fields[i], fs.scores_[i]))
5 # plot the scores
6 plt.bar([fields[i] for i in range(len(fs.scores_))], fs.scores_)
7 plt.show()
```

```
Feature bpm: 94.418302
Feature ibi: 161.422435
Feature sdn: 26.929492
Feature sds: 11.346253
Feature rmss: 7.110362
Feature pnn20: 19.999804
Feature pnn50: 0.838868
Feature hr_max: 28.252391
Feature sd1: 29.413127
Feature sd2: 123.773385
Feature s: 159.903890
Feature sd1/sd2: 131.737579
Feature breathingrate: 116.195230
```



The Scores of the features can be observed above.

The target variable/ predicted variable is label encoded by replacing values of 'N' with '0' and values of 'A' with '1'.

Model Building

Now that the data is transformed, we are going to train the K nearest neighbour classifier model. This can be observed below.

```
1 from sklearn.neighbors import KNeighborsClassifier
2
3 knn = KNeighborsClassifier()
4 knn.fit(x_train, y_train)
5 # evaluate the model
6 yhat = knn.predict(x_test)
7 # evaluate predictions
8 accuracy = accuracy_score(y_test, yhat)
9 print('Accuracy: %.2f' % (accuracy*100))
10 y_pred=knn.predict(x_test)
11 print('MSE:',np.sqrt(mean_squared_error(y_test,y_pred)))
12 print('f1_score:',f1_score(y_test,y_pred))
```

```
Accuracy: 90.71
MSE: 0.304724700110022
f1_score: 0.5962732919254659
```

As we can see the model's accuracy is 90.71% but it varies when trained at different times. The highest accuracy achieved was 93% and the f1_score ranged from 0.5 to 0.61.

Model Analysis

--->CLASSIFICATION REPORT=>

	precision	recall	f1-score	support
0	0.93	0.96	0.95	623
1	0.60	0.43	0.50	77
accuracy			0.91	700
macro avg	0.77	0.70	0.72	700
weighted avg	0.90	0.91	0.90	700

--->CONFUSION MATRIX=>

```
[[601  22]
 [ 44  33]]
```

In the classification report the class values '0' and '1' stand for 'N' normal reading and 'A' AF episode respectively. Due to imbalance of the data we can observe low 'f1_score' for class 'A' i.e., AF episode.

Saving the KNN Model, L1 Normalizer, ANOVA Transformer

The KNN Model, L1 Normalizer, ANOVA Transformer models can be saved using pickle module as shown.

```
import pickle as pkl
pkl.dump(knn,open('knn_model.pkl','wb'))
pkl.dump(Data_normalizer,open('normalizer.pkl','wb'))
pkl.dump(fs,open('ANOVA.pkl','wb'))
```

Evaluation Script

The following script can be used to evaluate the model using three pre-trained model files which are:

- **KNN model** saved as Pickle file
- **L1 normaliser** saved as Pickle file
- **ANOVA transformer** saved as Pickle file

The code for the evaluation can be observed below.

```
import pickle
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, f1_score

# Loading model and normalizer
model = pickle.load(open('knn_model.pkl', 'rb'))
normalizer = pickle.load(open('normalizer.pkl', 'rb'))
fs = pickle.load(open('ANOVA.pkl', 'rb'))

# Loading dataset
try:
    df = pd.read_csv('ECG_testing.csv')
    print('Dataset Loaded')
except:
    print('Check File/ path')

x = xx_noise.iloc[:, :13]
y = xx_noise.iloc[:, -1:]

# Label encoding
y = y.replace('N', 0)
y = y.replace('A', 1)

# Data Normalization
normalized_data = normalizer.transform(x)

# ANOVA Transformation
anova_data = fs.transform(normalized_data)

# Predicting values
y_pred = model.predict(anova_data)

# checking scores
print('F1_score is :', f1_score(y.values, y_pred))
print('accuracy is:', accuracy_score(y.values, y_pred)*100)
```

Output is the f1_score of the model with test_data and the accuracy_score.