A Report On The Analysis Of Random Forest Classifier

Mohammed Tahmid Hossain

Department Of Computer Science

Independent University Bangladesh

Dhaka, Bangladesh

1820228@iub.edu.bd

***Abstract*— In this report, I have discussed about the result of Random Forest Classifier model. I have used this model to train the data set from available from PhyNetLab. After studying the data, I figured it out that it was a classification problem. I have done data pre-processing , data visualization and trained the model with random forest classifier. This model gave a very good accuracy on the data set and after doing cross validation, I concluded that the accuracy of this model is correct for this data set.**

***Keywords*—Random Forest, Classification, Decision Trees**

I. Introduction

Random Forest is one of the most popular and powerful machine learning algorithm that uses supervised machine learning to train data. It is capable of doing both regression and classification problems. Random forest algorithm creates multiple decision trees to predict the outcome of a problem precisely. It is a group of un-pruned regression or classification trees that are created from choosing random selections of the data. This model predicts by taking the majority vote for classification problem and average value for regression problem.

Random forest has some advantages over other machine learning algorithms. It tends to be more accurate without overfitting the data, it handles missing values and maintains the accuracy for missing values. It also works accurately with huge datasets. It uses techniques like *Bagging*, *Bootstrapping* in the decision trees where the final predicted value is the average of all the values. A single decision tree might happen to overfit our model, but if we take all the decision trees into consideration, we will get an accurate prediction which won't overfit our model. Random Forest is a very versatile, robust model and it can work with high dimensional datasets.

II. Data Preprocessing

After appending all the datasets to a single dataframe, I studied the data and tried to understand the structure. As I saw the dataset was continous rather than discrete, I understood that it was a classification problem. I did data visualization and I looked at the distribution of my data based on my target column *(Pos)* .

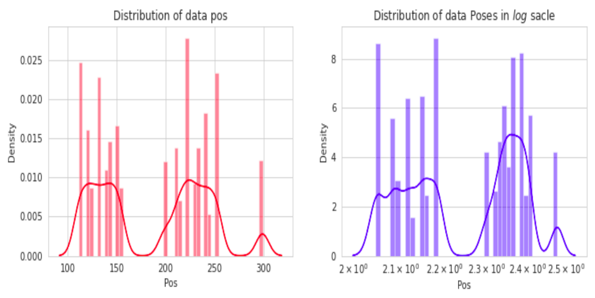


Figure 1: Distribution of Data

From the distribution, we can understand about the density of the classes in our data.

Random Forest classifier model handles missing values better than other model. After studying the data visualization, I did some data cleaning based on the feature diagram plotting. Data Preprocessing is very important when it comes to training different machine learning models. If data preprocessing is not done, our model can overfit our data. Data Preprocessing is basically data cleaning, which helps us to get rid of redundant data, noises from our dataframe.

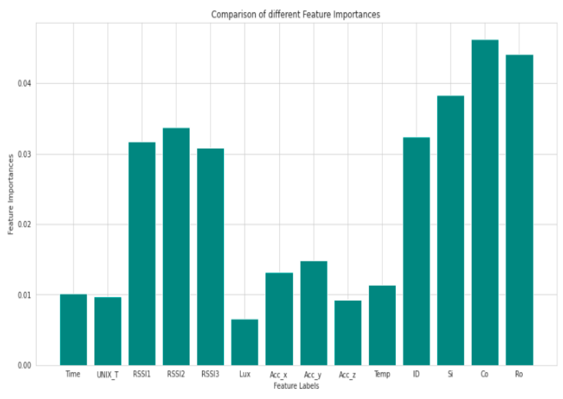


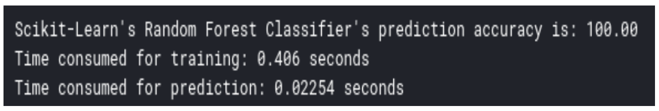
Figure 2: Feature Importance

There are many steps of data preprocessing. Encoding, Splitting, Feature scaling, Dropping etc. In this model, after studying the dataframe, I found that I have a data column which is a non integer type data (Object). Rest of the data was integer type. Hence, I did label encoding to convert the object data type to integer for my convenience. After that I did further data preprocessing and split my datasets into train and test parts. I also did feature scaling to standardize the independent variables of my dataframe within a range. This will help me to easily compare my data. I also checked if my dataframe had any missing values. The last part of data preprocessing is splitting the dataset *(test-train-split).* I used 80% of my data for training and the remaining 20% for testingAfter doing this part, I trained my model using the Random Forest Classifier and I found that my model overfits my data. I did not drop any columns at first but since my model was overfitting, I did some further preprocessing by dropping some irrelevant and redundant data columns from my dataframe. This is another method of preprocessing which is done when a model overfits our dataframe. After dropping the irrelevant columns, when we train our dataframe again, our model will be trained differently and this time it gave an accurate prediction. As I visualized the datasets with a feature diagram, from here I set the range of the datas *(min: 0.012, max: 0.04)* which I will use and drop the rest to solve our overfitting problem.

III. Justification

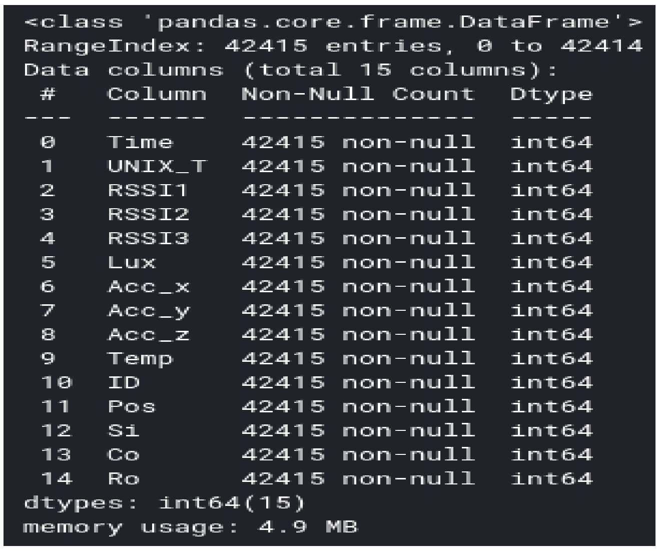
Overfitting is a very common problem in machine learning. It basically happens when a model learns about the noises in the training data to the extent that it negatively impacts the model performance. When overfitting occurs, there are many ways to get rid of it.

In my first attempt, my model was also overfitting my data. After studying the feature diagram of my dataframe, I set the ranges of my data manually that I will use to train the model and drop the rest of the columns from my dataframe. Feature diagram is very important in this regard. Without feature diagram, I cannot arbitrarily decide which columns I use and which columns I drop without visualizing the feature importance of our dataframe. For better understanding, you can see below my model was overfitting my dataframe when I did not drop any columns. The accuracy was 100%.



```
Scikit-Learn's Random Forest Classifier's prediction accuracy is: 100.00
Time consumed for training: 0.406 seconds
Time consumed for prediction: 0.02254 seconds
```

Figure 3: Overfitting



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42415 entries, 0 to 42414
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Time     42415 non-null  int64
 1   UNIX_T   42415 non-null  int64
 2   RSSI1    42415 non-null  int64
 3   RSSI2    42415 non-null  int64
 4   RSSI3    42415 non-null  int64
 5   Lux      42415 non-null  int64
 6   Acc_x    42415 non-null  int64
 7   Acc_y    42415 non-null  int64
 8   Acc_z    42415 non-null  int64
 9   Temp     42415 non-null  int64
 10  ID       42415 non-null  int64
 11  Pos      42415 non-null  int64
 12  Si       42415 non-null  int64
 13  Co       42415 non-null  int64
 14  Ro       42415 non-null  int64
dtypes: int64(15)
memory usage: 4.9 MB
```

Figure 4: Dataframe before dropping column

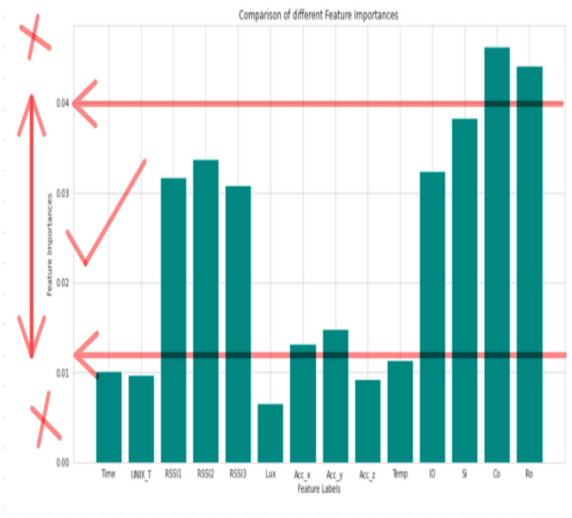
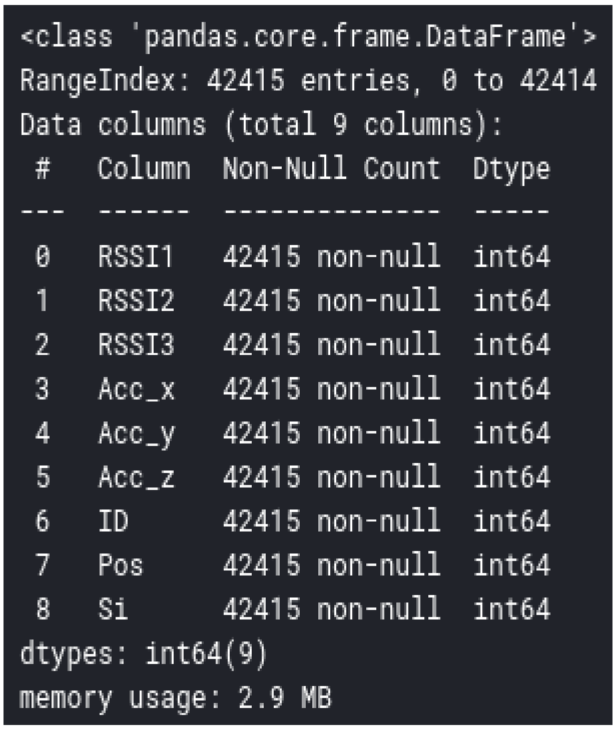


Figure 5: Dropping Columns

After I realized that my model is overfitting my dataframe, I looked at the feature diagram to study my dataframe. After studying the feature importance, I set the range of the data that I wanted to take manually. I set the minimum value as 0.015 and the maximum value as 0.04. As a result, the data that is less than 0.015 and greater than 0.04 was treated as anomaly and those were dropped from the dataframe.

As I set the range (0.012-0.04) hence Time, UNIX_T, Lux & Temp was dropped as the values of those columns are less than the minimum value. Again on the other hand, Co and Ro column was dropped as it crossed the maximum value.

After dropping the columns, this is how my datafrane and feature importance looks like



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42415 entries, 0 to 42414
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RSSI1   42415 non-null  int64
 1   RSSI2   42415 non-null  int64
 2   RSSI3   42415 non-null  int64
 3   Acc_x   42415 non-null  int64
 4   Acc_y   42415 non-null  int64
 5   Acc_z   42415 non-null  int64
 6   ID      42415 non-null  int64
 7   Pos     42415 non-null  int64
 8   Si      42415 non-null  int64
dtypes: int64(9)
memory usage: 2.9 MB
```

Figure 6: Dataframe after dropping columns

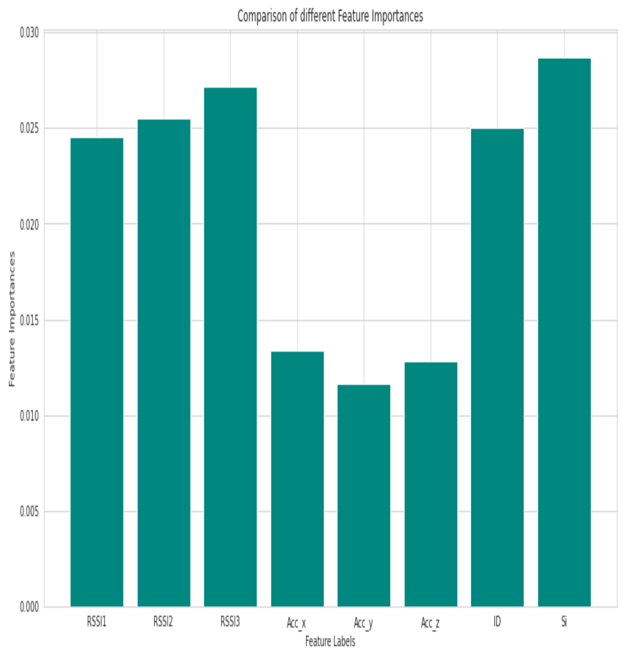
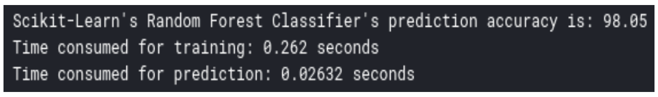


Figure 7: Feature Importamce after dropping columns

Now after I dropped those columns, I again ran my model and this time I model did not overfit my dataset



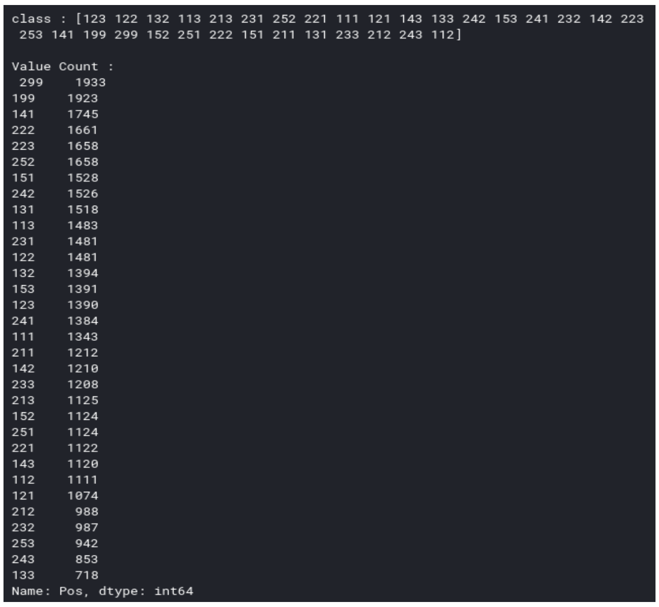
```
Scikit-Learn's Random Forest Classifier's prediction accuracy is: 98.05
Time consumed for training: 0.262 seconds
Time consumed for prediction: 0.02632 seconds
```

Figure 8: Correct Accuracy

Here, I corrected the problem of overfitting by setting range dropping some columns And since my model is giving me a really good accuracy, 98%, I can conclude that random forest classifier model is doing a good really good job predicting my datasets.

III. Interpretation

After studying the dataframe, I realized that this is a classification problem for which I used Random Forest Classifier instead of decision trees to train the data.



```
class : [123 122 132 113 213 231 252 221 111 121 143 133 242 153 241 232 142 223
 253 141 199 299 152 251 222 151 211 131 233 212 243 112]

Value Count :
 299    1933
199    1923
141    1745
222    1661
223    1658
252    1658
151    1528
242    1526
131    1518
113    1483
231    1481
122    1481
132    1394
153    1391
123    1390
241    1384
111    1343
211    1212
142    1210
233    1208
213    1125
152    1124
251    1124
221    1122
143    1120
112    1111
121    1074
212     988
232     987
253     942
243     853
133     718
Name: Pos, dtype: int64
```

Figure 9: Data Classification

Decision trees does seem like a perfect classifier since it takes a series of yes/no decisions asked about our data which eventually led us to the prediction. But that won't give us the correct accuracy as the tree don't reach to the max depth. Random Forest model is made up of multiple decision trees. Hence instead of just finding the mean of the prediction of trees, it uses random sampling (Bootstrapping) and random subset of features.

Random forest combines thousands of decision trees together and trains each tree on a slightly unique set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the model is made by averaging the predictions of each individual tree.

We can easily understand the reliability of random forest model due to it's features. In a decision tree, the prediction was entirely based on the data. But the dataset can contain noise in addition to the data. As the decision tree is basing the prediction solely on the data available, it will have a high chance of getting swayed by irrelavant information. The model can come up with altering prediction from the same dataset.

The solution to this problem is not to rely solely on any INDIVIDUAL data but relying on a pool of majority votes of each tree. Due to bootstrapping, the noises in the data will get cancelled out. This is why Random Forest uses a multiple number of decision trees and combines them into a random forest, instead of using a tree for the entire data.

In my model, after getting rid of the overfitting problem, it gave a very good accuracy of 98%. This means that my model created a forest of hundreds and thousands of decision trees and used those decision trees to predict the dataset with bootstrapping or random sampling. As the noises gets cancelled out in this process, my dataset does not get mixed with irrelevant information. Lastly, after combining all the decision trees, my model takes the majority vote in each tree and predicts my data accurately. And hence it gives an accuracy of 98%.

IV. Evaluation

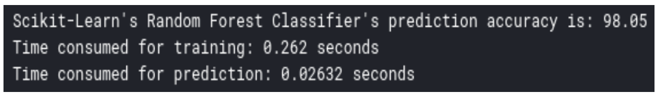
After training our model, the final task is to evaluate the model and to check how good or bad it actually performs.

There are certain factors that help us to identify and evaluate the performance of a model. They are:

- Accuracy
- Precision
- Recall
- F1 Score
- Confusion Matrix

Accuracy: Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Sometimes we might think that higher the accuracy the better the performance. Accuracy is a great measure but only when the dataset is symmetric where values of false positive and false negatives are almost same. Therefore, I have to look at other parameters to evaluate the performance of your model. For our model, we have got 98.05 % means our model is approx. 98.05% accurate.

Accuracy = TP+TN/TP+FP+FN+TN



```
Scikit-Learn's Random Forest Classifier's prediction accuracy is: 98.05
Time consumed for training: 0.262 seconds
Time consumed for prediction: 0.02632 seconds
```

Figure 9: Accuracy

Precision – We calculate precision by finding the ratio of correctly predicted positive observations to the total predicted positive observations. If we have high precision value, this means we have low false positive rate. We have got 0.999 precision which is pretty good

Precision = TP/TP+FP

Recall – We calclate recall by calculating the ratio of correctly predicted positive observations to the all observations in actual class. We have got recall of 0.99 which is good for this model as it's above 0.5.

Recall = TP/TP+FN

F1 score - F1 Score is the average of Precision and Recall values. It takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, In our case, F1 score is 0.99.

F1 Score = 2(Recall * Precision) / (Recall + Precision)*

Accuracy	Precision	Recall	Mean	Deviation	F1-Score
98% (approx)	0.999	0.999	0.969	0.001	0.999

One thing that should be noted that all the values of my precision, recall and F1 scores are exactly identical. The reason of such This means that my model is predicting correctly but there might be some mistakes in data preprocessing which could affect in these values.

Confusion Matrix: A confusion matrix is an NxN matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a clear view of how well our classification model is performing or what kinds of errors it is making

IV. Conclusion

After cross validating my model, I found that my model did gave some accurate prediction. Hence, theoretically, I would say that my model is correct. But practically, there are some issues with the preprocessing and data cleaning part because my precision, recall and F1-Score values were identical. I dropped some columns by setting the ranges manually, that might cause me to drop some important features for which this could happen. Apart from that, I can say that this model is working correctly theoretically.

V. References

1. *Pal, M. (2005). Random forest classifier for remote sensing classification. International journal of remote sensing, 26(1), 217-222.*
2. Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random forests and decision trees. International Journal of Computer Science Issues (IJCSI), 9(5), 272.
3. Xu, B., Guo, X., Ye, Y., & Cheng, J. (2012). An Improved Random Forest Classifier for Text Categorization. JCP, 7(12), 2913-2920.
4. Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. ISPRS Journal of Photogrammetry and Remote Sensing, 67, 93-104.