Machine Learning 1 Project Report: Recidivism Predictive Model

Team Members:

James Beck, Jia Mody

10/23/24

Table of Contents

Part 1 – Statement/Project Goal

According to the Bureau of Justice Statistics, approximately 68% of convicted criminals commit a second crime within three years of release from jail. This trend highlights the necessity for strategies to address recidivism as repeat offenders not only overburden the legal system but also pose security risks to communities. In this project, our goal is to develop a comprehensive risk assessment model that identifies characteristics and behavioral patterns of criminals at high risk of reoffending. Specifically we looked at socio-economic, psychological, and environmental factors.

Leveraging data-driven insights from this study, legal institutions would have the tools necessary to identify potential repeat offenders and allocate targeted resources for their rehabilitation.This would invariably enhance public safety while ensuring high risk offenders receive adequate support throughout rehabilitation. This model could also provide valuable insights for policymakers, allowing them to design more effective parole conditions and post-release programs.

Part 2 – Description of Dataset

Our dataset was published on the "Data.gov" website by the U.S. Department of Justice. The dataset includes a total of 25,835 instances and 49 attributes excluding the class attributes. There are four potential class attributes for our model: "Recidvism_Arrest_Year1," "Recidvism_Arrest_Year2," "Recidvism_Arrest_Year3," and "Recidivism_Within_3years". Because a class variable that combines the information from each of these variables would make the most sense for our project, we will combine these four variables into a qualitative variable consisting of the four labels "1", "2", "3", and "Never". The label "Never" would correspond to

a criminal who was not arrested within 3 years after release from jail, “1” would correspond to a criminal arrested within one year of release, “2” a criminal arrested within two years, and so on. 10931 convicts did not commit a second crime within three years, 7724 committed a second crime within one year, 4567 within two, and 2613 within three. Therefore, convicts who commit a second crime appear more likely to do so soon after release from jail, creating a right skew. Below is an exhaustive list of the attributes:

	Attribute	Description	# Missing Values
1	ID	Number of person (1-25,825)	0
2	Gender	Male or Female	0
3	Race	Black or White	0
4	Age_at_Release	Age when released from jail	0
5	Residence_PUMA	Public Use Microdata Area: number correspond to a place of residence	0
6	Gang_Affiliated	Is the person affiliated with a gang?	3,167
7	Supervision_Risk_ Score_First	Supervision risk level in number format. Base on severity of first crime committed	475

8	Supervision_Level_First	Supervision risk level in text format. Base on severity of first crime committed	1,720
9	Education_Level	Highest level of education	0
10	Dependents	Number of people dependent on inmate's income	0
11	Prison_Offense	Type of offense	3, 277
12	Prison_Years	Years in prison	0
13	Prior_Arrest_Episodes_Felony	Number of prior arrests	0
14	Prior_Arrest_Episodes_Misd	Number of arrests due to misdemeanors	0
15	Prior_Arrest_Episodes_Violent	Number of arrests due to violence	0
16	Prior_Arrest_Episodes_Property	Number of arrests due to property damages	0
17	Prior_Arrest_Episodes_Drug	Number of arrests due to drug usage	0

18	Prior_Arrest_Episodes_PPViolationCharges	Number of arrests due to protective order violations?	0
19	Prior_Arrest_Episodes_DVCharges	Have there been arrests due to domestic violence?	0
20	Prior_Arrest_Episodes_GunCharges	Have there been arrests due to gun charges	0
21	Prior_Conviction_Episodes_Felony	Number of convictions due to felony	0
22	Prior_Conviction_Episodes_Misd	Number of convictions due to misdemeanors	0
23	Prior_Conviction_Episodes_Viol	Number of convictions due to violence	0
24	Prior_Conviction_Episodes_Prop	Number of convictions due to property damages	0
25	Prior_Conviction_Episodes_Drug	Number of convictions due to drug usage	0

26	Prior_Conviction_Episodes_PPViolationCharges	Have there been convictions due to protective order violations?	0
27	Prior_Conviction_Episodes_DomesticViolenceCharges	Have there been convictions due to domestic violence?	0
28	Prior_Conviction_Episodes_GunCharges	Have there been convictions due to gun charges?	0
29	Prior_Revocations_Parole	Have there been previous violations of parole terms?	0
30	Prior_Revocations_Probation	Have there been previous violations of probation terms?	0
31	Condition_MH_SA	Has there been substance abuse/bad mental health?	0
32	Condition_Cog_Ed	Is there a cognitive or educational condition?	0
33	Condition_Other	Are there any other conditions?	0

34	Violations_ElectronicMonitoring	Have there been breaches of electronic monitoring rules?	0
35	Violations_Instruction	Have there been breaches of instruction rules?	0
36	Violations_FailToReport	Have there been failures to report as required?	0
37	Violations_MoveWithoutPermission	Has the inmate moved without permission?	0
38	Delinquency_Reports	Number of delinquency reports	0
39	Program_Attendances	Number of programs attended	0
40	Program_UnexcusedAbsences	Number of unexcused absences from program	0
41	Residence_Changes	Number of place of residence changes	0
42	Avg_Days_per_DrugTest	Days between drug tests, on average	6,103

43	DrugTests_THC_Positive	Number of drug tests that testedTHC positive	5,172
44	DrugTests_Cocaine_Positive	Number of drug tests that tested cocaine positive	5,172
45	DrugTests_Meth_Positive	Number of drug tests that tested meth positive	5,172
46	DrugTests_Other_Positive	Number of drug tests that tested positive for other drugs	5,172
47	Percent_Days_Employed	Percentage of 365/366 days employed	462
48	Jobs_Per_Year	Number of jobs worked per year	808
49	Employment_Exempt	Is the inmate exempt from employment?	0

Figure 1. Data Table

Part 3 – Pre-Processing

Pre-Processing was done using WEKA 3.8.6 and python scripts in Jupyter Notebook and VS Code.

Part 3.1 – Replace Missing Values

First, we began by filling in missing values. We did not remove any attributes because they all had <70% missing values. Because our quantitative attributes were heavily skewed, we used the attribute's median to fill in the missing values rather than the mean. This is because medians are more robust against outliers than means. Additionally, most of our quantitative attributes were discrete whole numbers, so it did not make sense to fill in missing values with decimal means. However, by default, WEKA fills in missing values with means for quantitative attributes. Furthermore WEKA lacks that ability to fill in missing values with median. Therefore, in order to fill in missing values with the median, we developed a Python script to compute the median for each quantitative attribute in our dataset and replace the missing values accordingly. This script ensures that the filled in missing values make sense with the distributions of their respective attributes.

The python script we used is shown in *Figure 2* below.

```
import csv
dataset = []
#Filling in missing values:
with open("NIJ_s_Recidivism_Challenge_Full_Dataset.csv", mode='r')as file:
    fileReader = csv.reader(file)
    dct = {6:[], 41:[], 42:[], 43:[], 44:[], 45:[], 46:[], 47:[]}
    for i, line in enumerate(fileReader):
        dataset.append(line)
        if i == 0: continue
        for key in dct:
```

```
        if line[key] == "": continue
        dct[key].append(float(line[key]))
medianDct = {}

for key in dct:
    dct[key].sort()
    medianDct[key] = dct[key][len(dct[key])//2]

for rowNum, row in enumerate(dataset):
    for col, val in enumerate(row):
        if val == "" and col in medianDct:
            dataset[rowNum][col] = medianDct[col]
```

Figure 2. Python Script Demonstrating Missing Data Replacement

We then used the "ReplaceMissingValues" filter to fill in missing values for qualitative variables with the mode. All missing values being filled, we re-uploaded the dataset to WEKA for further analysis.

Part 3.2 – Create Derived Class

In our original dataset, there were four potential class attributes: "Recidivism_Arrest_Year1," "Recidivism_Arrest_Year2," "Recidivism_Arrest_Year3," and "Recidivism_Within_3Years." Each of these attributes represent specific arrest outcomes

following an offender's release. Because we cannot have multiple class variables, we decided to create a single qualitative class attribute that captures the information from each of these four variables.

We combined these variables into a qualitative variable with the following four labels:

- "1" – Arrested within 1 year of release
- "2" – Arrested within 2 years of release
- "3" – Arrested within 3 years of release
- "Never" – Not arrest within 3 years of release

The python script we used is shown in *Figure 3*.

```
#Combining classes:
with open('CombinedClass.csv', mode='w', newline='') as file:
    training = dataset[0].pop()
    Year3 = dataset[0].pop()
    Year2 = dataset[0].pop()
    Year1 = dataset[0].pop()
    within3 = dataset[0].pop()
    dataset[0].append("Years_Until_Recidivism")

    for i in range(len(dataset)-1):
        training = dataset[i+1].pop()
        Year3 = dataset[i+1].pop()
        Year2 = dataset[i+1].pop()
        Year1 = dataset[i+1].pop()
        within3 = dataset[i+1].pop()

        combinedVal = "Never"
        if Year1 == "true": combinedVal = "1"
        if Year2 == "true": combinedVal = "2"
        if Year3 == "true": combinedVal = "3"
        dataset[i+1].append(combinedVal)
```

```
    writer = csv.writer(file)
    writer.writerows(dataset)
```

Figure 3. Python Script Demonstrating Derived Class Creation

Part 3.3 – Normalize Data

To ensure that no attribute was over/underrepresented in our classification model, we needed to normalize the data. For this step, we used WEKA's "Normalize" filter which scales all quantitative attributes into the range [0, 1].

However, one attribute named "Residence_PUMA" did not make sense to normalize as it contained residency IDs. Because WEKA's "Normalize" filter normalizes all quantitative variables, we had to convert "Residence_PUMA" to a nominal attribute. To do so, we used WEKA's "NumericToNominal," which successfully converted "Residence_PUMA" to a nominal attribute. This step would successfully exclude "Residence_PUMA" from the normalization process.

We then applied the "NumericToNominal" filter, successfully normalizing the remaining quantitative attributes.

Part 3.4 – Changing Necessary Values from Nominal to Numeric

While preprocessing, we discovered that several quantitative attributes were marked "nominal" because they included values of the format "# or more" (e.g., "5 or more"). For example, "Prior_Arrest_Episodes_Felony" contained values from 0 to 9 and then values of "10 or more". This caused WEKA to interpret the entire column as categorical. For simplicity, we

decided to replace these values with numbers (i.e. “5 or more” would be replaced with “5”) in order to make these attributes quantitative in WEKA.

To do this, we imported the dataset into Google Sheets and manually edited the affected columns. Specifically, we removed the “or more” suffix from each instance, leaving only the numeric portion (i.e., converting “5 or more” to “5”). This ensured that the entire column was properly recognized as numeric. With these changes made, we re-exported the modified dataset and uploaded it to WEKA.

Part 3.5 – Split final dataset into training and test dataset

After preprocessing our data, we needed to split it into train, validation, and test sets. Our dataset contained 25,825 instances, so we decided to create a 70/15/15 split (70% for training, 15% for testing, and 15% for validating). Additionally, because our dataset is unbalanced, we needed to ensure that each section had a sample that was representative of the class distribution. To do this we utilized a stratified random sampling method within Google CoLab. After this, however, we realized that our code had errors when creating the validation set. Since the train dataset was stratified, the number of instances in it did not match the number of instances in our dataset. We then decided to stratify the validation set based on the training set. We thought that this would work, but upon further inspection we noticed that this method does not result in an exact 70/15/15 split, since the validation set is stratified for 15% based on the 85% training set. We then decided to stratify the validation set based on the testing set. For this, we had to change the first split to be 70/30. From there, we stratified 50% of the 30% for validation and the other half for train.

The script used is shown in *Figure 4* Below

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/ML/Normalized.csv')

from google.colab import drive

drive.mount('/content/drive')

from sklearn.model_selection import train_test_split

train, remaining = train_test_split(df, test_size=0.30, stratify=df.iloc[:, -1])

val, test = train_test_split(remaining, test_size=0.50, stratify=remaining.iloc[:, -1])

train.to_csv('train.csv', index=False)

!cp train.csv /content/drive/MyDrive/ML

val.to_csv('val.csv', index=False)

!cp val.csv /content/drive/MyDrive/ML

test.to_csv('test.csv', index=False)

!cp test.csv /content/drive/MyDrive/ML
```

Figure 4: Python Script Demonstrating Train Test Validation Splits

Down the line we had issues utilizing the validation set in WEKA, so we decided to do a train-test split of 70/30 without validation. The updated script is seen in *Figure 5* below.

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/ML/Normalized.csv')

from google.colab import drive

drive.mount('/content/drive')

from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size=0.30, stratify=df.iloc[:, -1])
```

```
train.to_csv('train.csv', index=False)
!cp train.csv /content/drive/MyDrive/ML
test.to_csv('test.csv', index=False)
!cp test.csv /content/drive/MyDrive/ML
```

Figure 5: Python Script Demonstrating Train Test Splits

Part 4 – Attribute Selection Algorithms & Model Classifiers Used

Part 4.1 Attribute Selection Algorithms:

To select potentially useful attributes, we used the following 4 attribute selection algorithms in addition to self selection:

- CorrelationAttributeEval with cutoff 0.1
- InfoGainAttributeEval with cutoff 0.25
- OneRAttributeEval with cutoff 43.95
- WrapperSubsetEval

These 5 methods gave us the following attribute selections:

#	CorrelationAttributeEval	InfoGainAttributeEval	OneRAttributeEval	WrapperSubsetEval	Self Selection
1	Percent_Days_Employed	Jobs_Per_Year	Jobs_Per_Year	Gang_Affiliated	DrugTests_Cocaine_Positive

2	Prior_Arrest_Episodes_PPViolationCharges	Percent_Days_Employed	Percent_Days_Employed	Prior_Arrest_Episodes_PPViolationCharges	DrugTests_Meth_Positive
3	Prior_Arrest_Episodes_Felony	Prior_Arrest_Episodes_PPViolationCharges	Gang_Affiliated	Prior_Conviction_Episodes_PPViolationCharges	Gang_Affiliated
4	Gang_Affiliated	Prior_Arrest_Episodes_Felony	Prior_Arrest_Episodes_PPViolationCharges	Violations_FailToReport	Prison_Years
5	Prior_Arrest_Episodes_Property	Gang_Affiliated	DrugTests_THC_Positive	Deliquency_Reports	Condition_Cog_Ed
6	Supervision_Risk_Score_First	Supervision_Risk_Score_First	Prior_Arrest_Episodes_Property	Percent_Days_Employed	Education_Level
7	Prior_Arrest_Episodes_Misd	DrugTests_THC_Positive	Prior_Arrest_Episodes_Felony	Jobs_Per_Year	Dependent

8	Prior_Conviction_Episodes_Misd	Prior_Arrest_Episodes_Property	Age_at_Release		Violations_Instruction
9	Prior_Conviction_Episodes_Prop	Age_at_Release	Prior_Conviction_Episodes_Prop		Percent_Days_Employed
10			Supervision_Risk_Score_First		

Figure 6: Table Demonstrating Attribute Selection tests and their Results

Part 4.2 Classifier Models

To construct our classification models, on each combination of attributes selected by our attribute selection algorithms, we used the following 4 classification algorithms:

- J48
- NaiveBayes
- OneR
- RandomForest

In WEKA, we created and tested all 20 models using K-Fold cross validation with 10 folds. The outputs for these models is shown below:

J48 – CorrelationAttributeEval Attributes

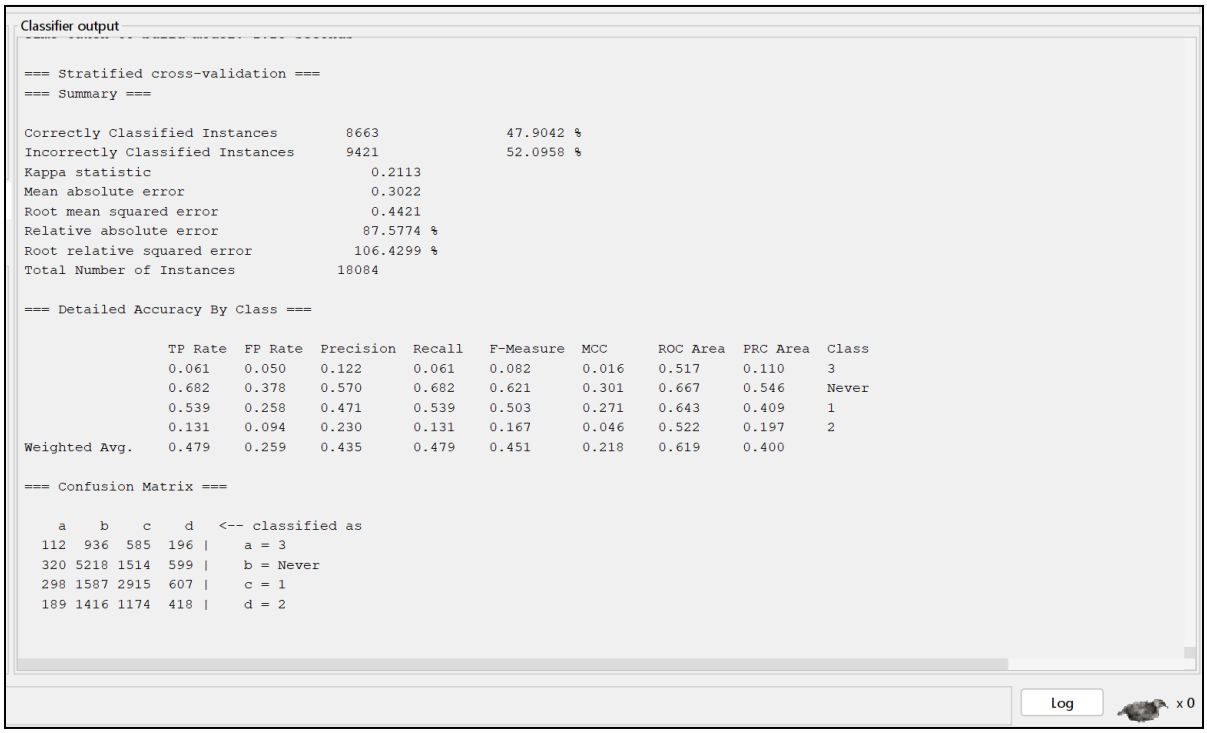
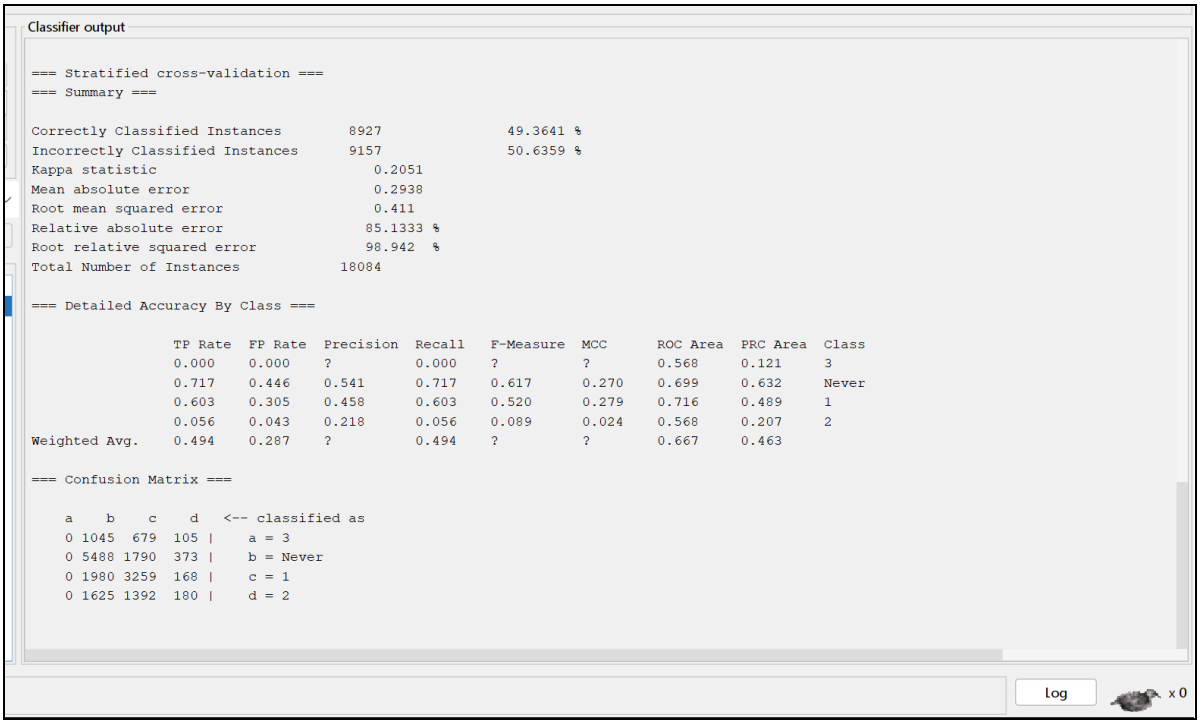


Figure 7: J48 CorrelationAttributeEval Correlation Result

Naive Bayes – CorrelationAttributeEval Attributes



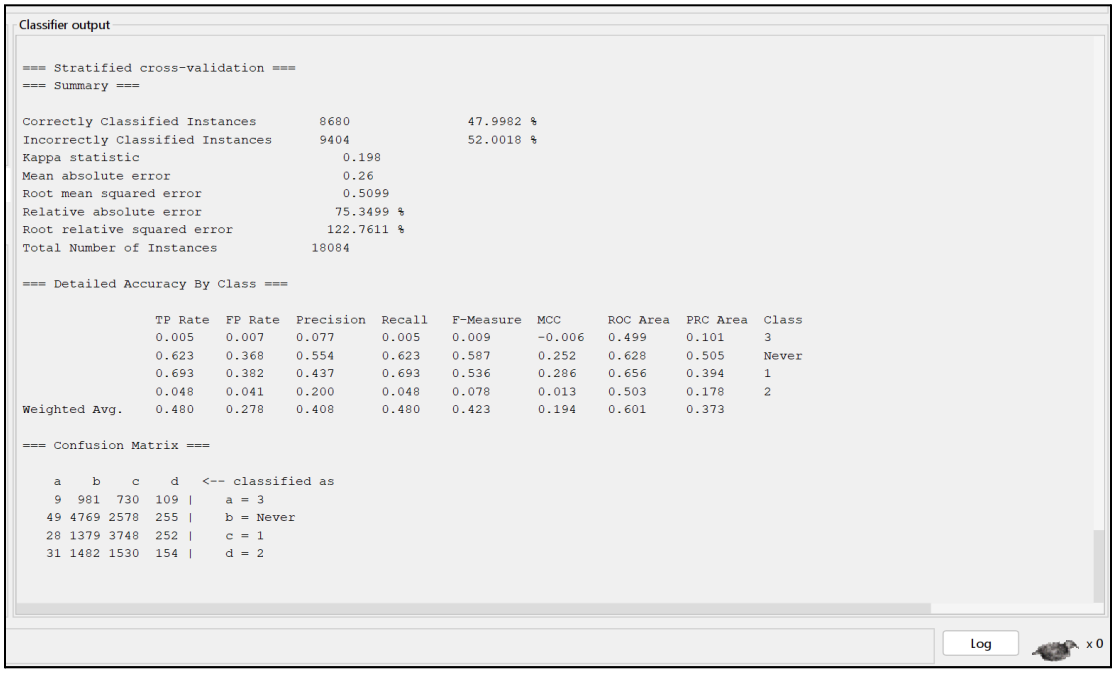
Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        8927               49.3641 %
Incorrectly Classified Instances      9157               50.6359 %
Kappa statistic                          0.2051
Mean absolute error                      0.2938
Root mean squared error                  0.411
Relative absolute error                 85.1333 %
Root relative squared error             98.942  %
Total Number of Instances            18084

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.000    0.000    ?          0.000    ?          ?        0.568     0.121     3
               0.717    0.446    0.541      0.717    0.617      0.270    0.699     0.632     Never
               0.603    0.305    0.458      0.603    0.520      0.279    0.716     0.489     1
               0.056    0.043    0.218      0.056    0.089      0.024    0.568     0.207     2
Weighted Avg.  0.494    0.287    ?          0.494    ?          ?        0.667     0.463

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
    0 1045  679  105 |    a = 3
    0 5488 1790  373 |    b = Never
    0 1980 3259  168 |    c = 1
    0 1625 1392  180 |    d = 2
```

Log x 0

Figure 8: NaiveBayes CorrelationAttributeEval Correlation Result

OneR – CorrelationAttributeEval Attributes



Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        8680               47.9982 %
Incorrectly Classified Instances      9404               52.0018 %
Kappa statistic                          0.198
Mean absolute error                      0.26
Root mean squared error                  0.5099
Relative absolute error                 75.3499 %
Root relative squared error            122.7611 %
Total Number of Instances            18084

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.005    0.007    0.077      0.005    0.009      -0.006   0.499     0.101     3
               0.623    0.368    0.554      0.623    0.587      0.252    0.628     0.505     Never
               0.693    0.382    0.437      0.693    0.536      0.286    0.656     0.394     1
               0.048    0.041    0.200      0.048    0.078      0.013    0.503     0.178     2
Weighted Avg.  0.480    0.278    0.408      0.480    0.423      0.194    0.601     0.373

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
    9  981  730  109 |    a = 3
   49 4769 2578  255 |    b = Never
   28 1379 3748  252 |    c = 1
   31 1482 1530  154 |    d = 2
```

Log x 0

Figure 9: OneR CorrelationAttributeEval Correlation Result

RandomForest – CorrelationAttributeEval Attributes

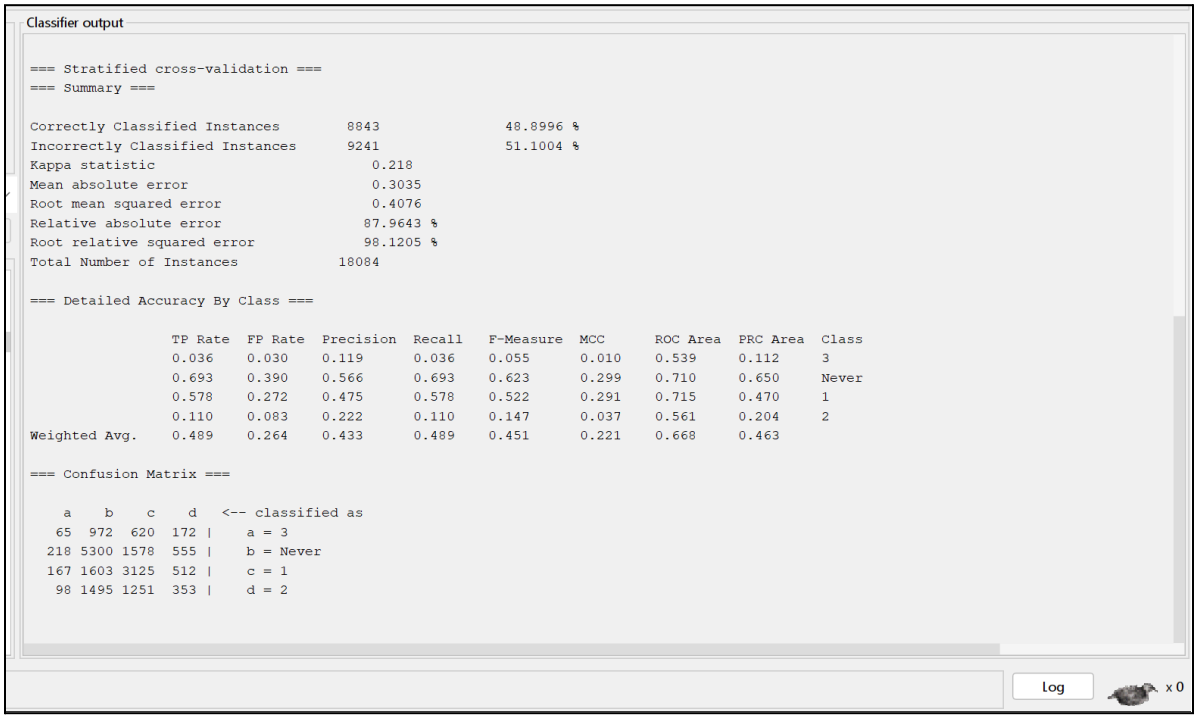


Figure 10: RandomForest CorrelationAttributeEval Correlation Result

J48 – InfoGainAttributeEval Attributes

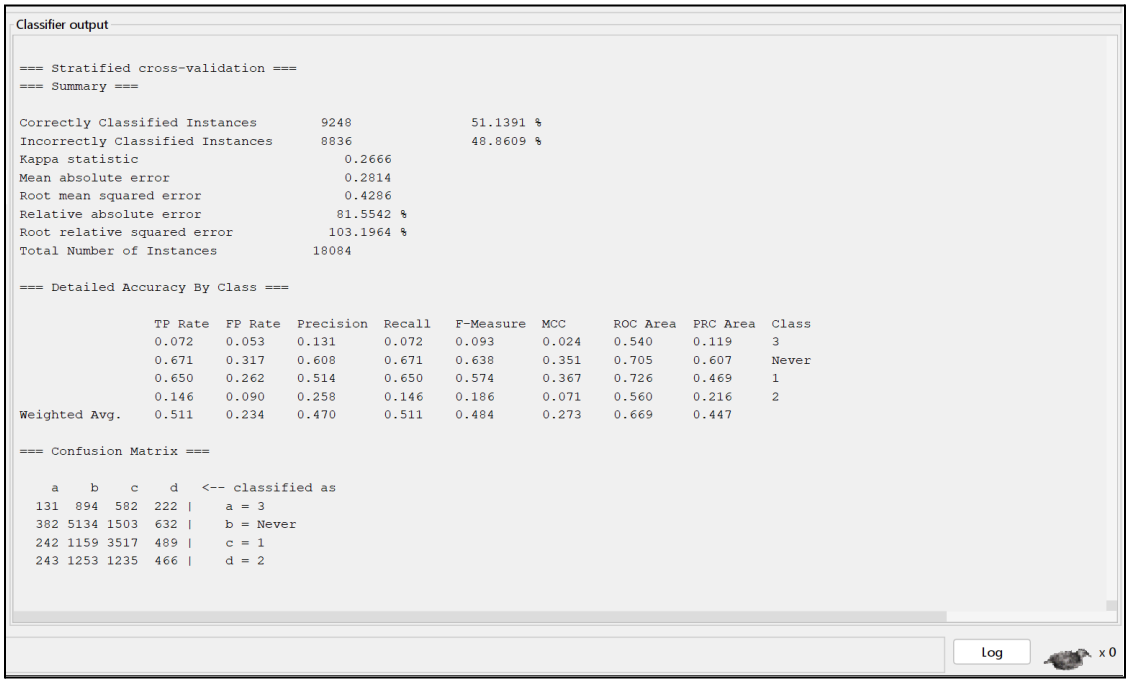


Figure 11: J48 InfoGainAttributeEval Correlation Result

Naive Bayes – InfoGainAttributeEval Attributes

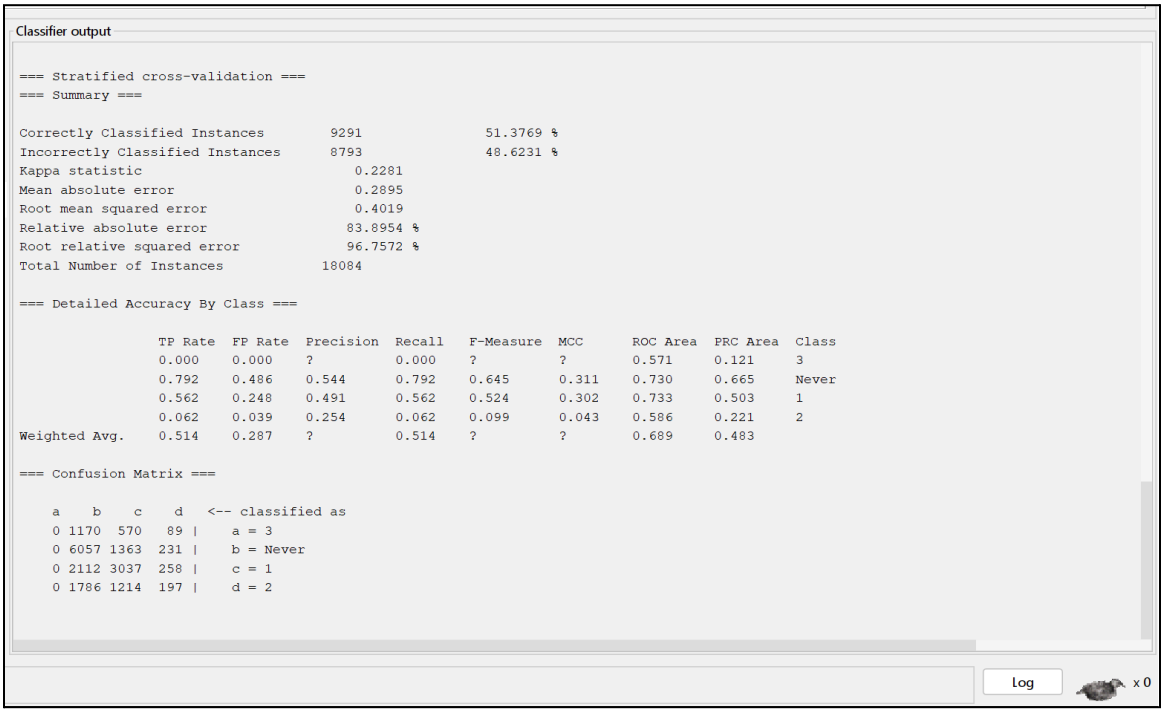


Figure 12: NaiveBayes InfoGainAttributeEval Correlation Result

One R – InfoGainAttributeEval Attributes

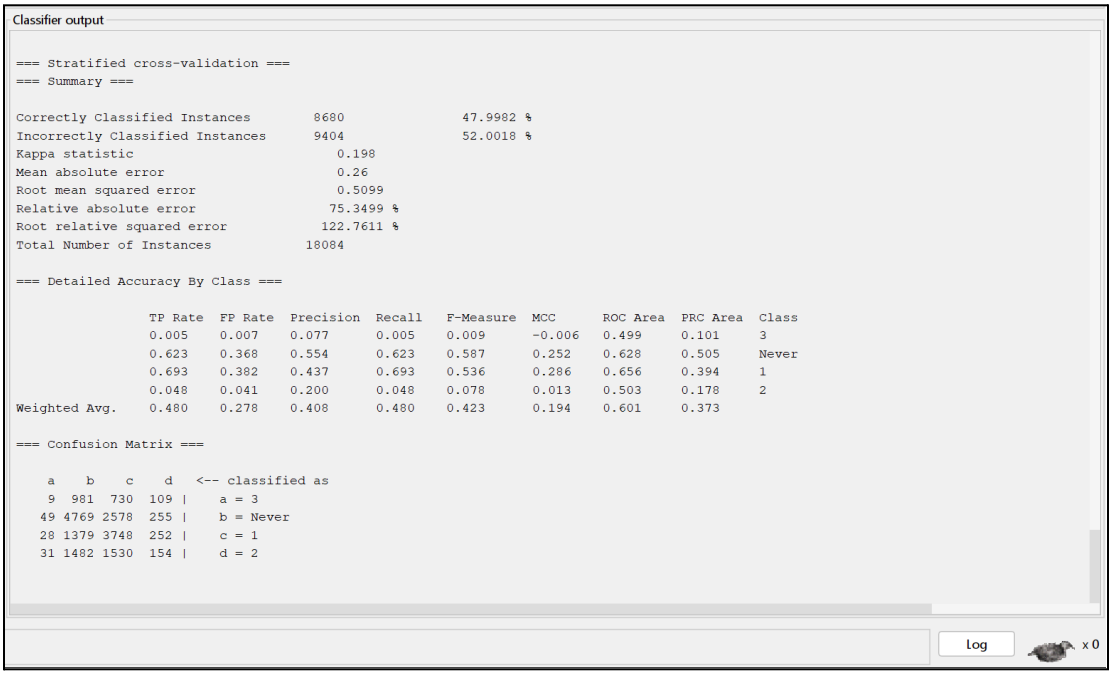


Figure 12: OneR InfoGainAttributeEval Correlation Result

RandomForest – InfoGainAttributeEval Attributes

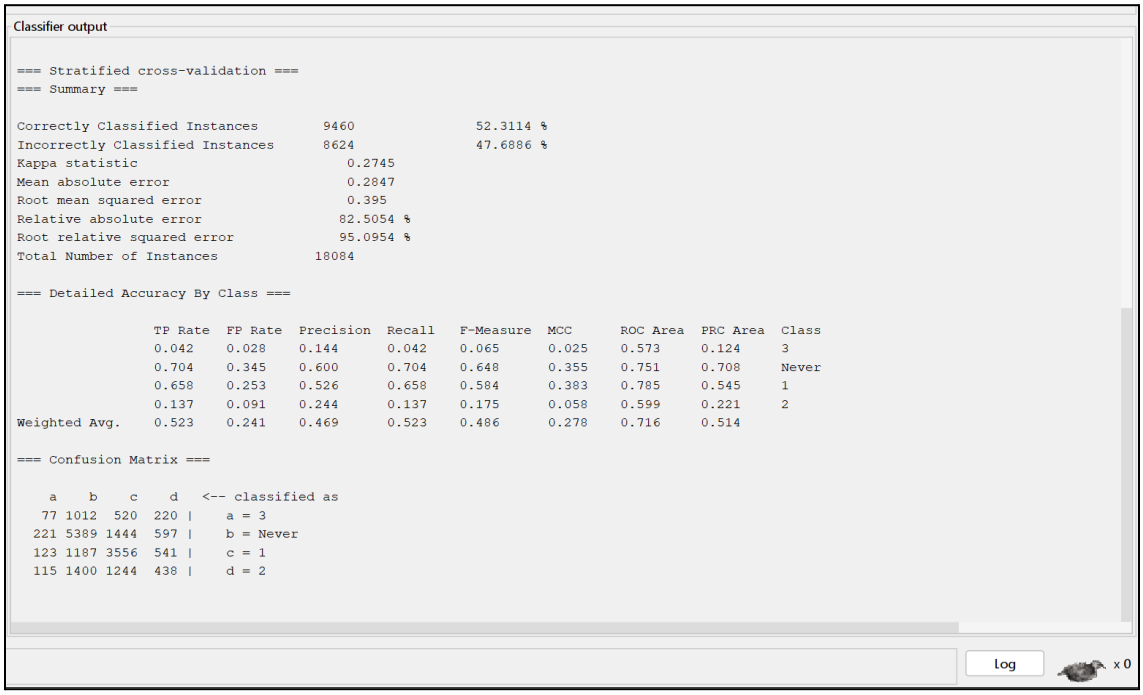


Figure 13: RandomForest InfoGainAttributeEval Correlation Result

J48 – OneRAttributeEval Attributes

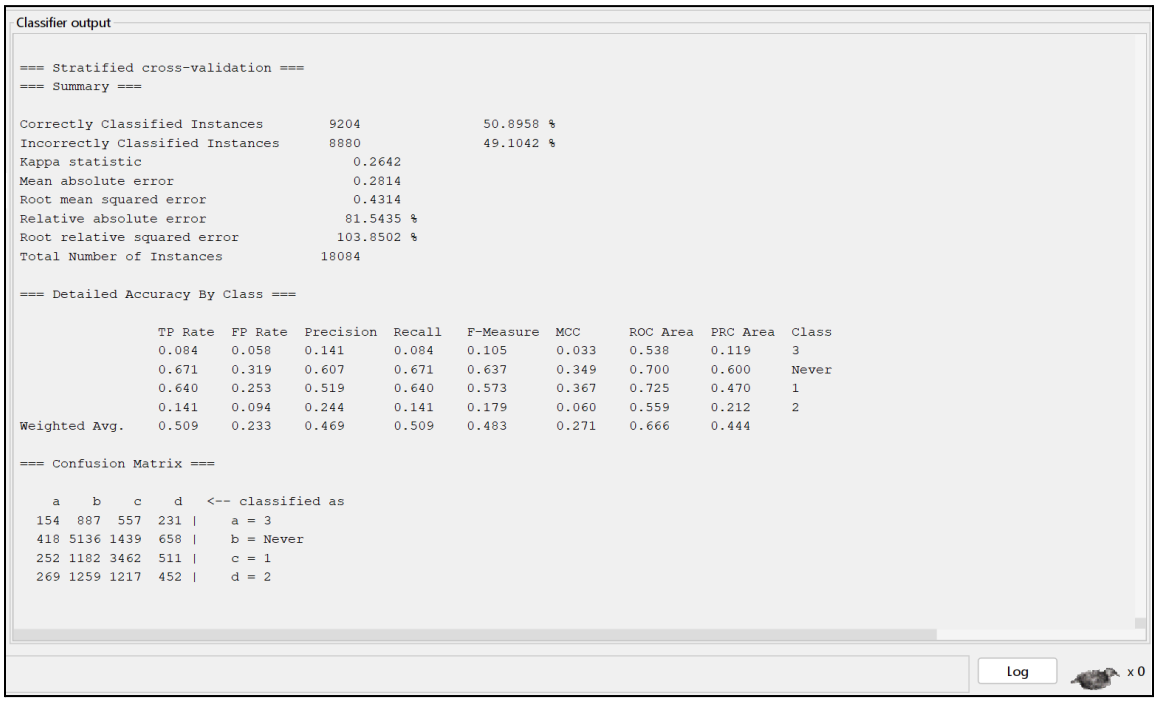


Figure 14: J48 OneRAttributeEval Correlation Result

Naive Bayes – OneRAttributeEval Attributes

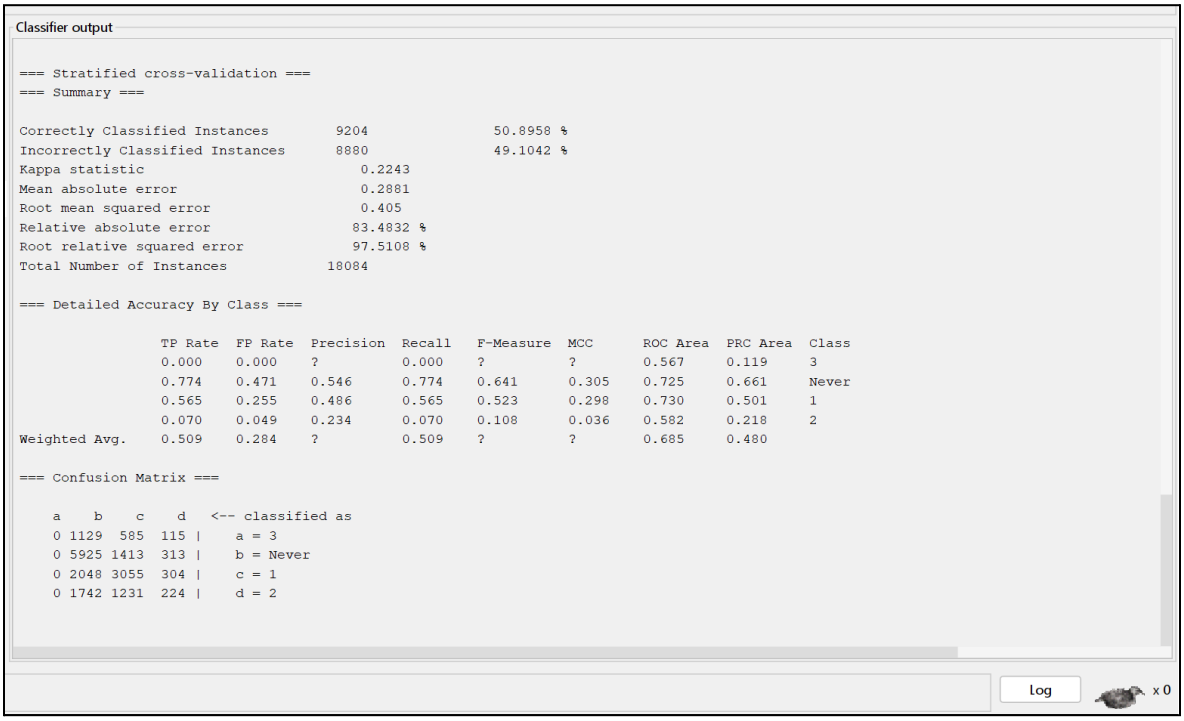


Figure 15: NaiveBayes OneRAttributeEval Correlation Result

OneR – OneRAttributeEval Attributes

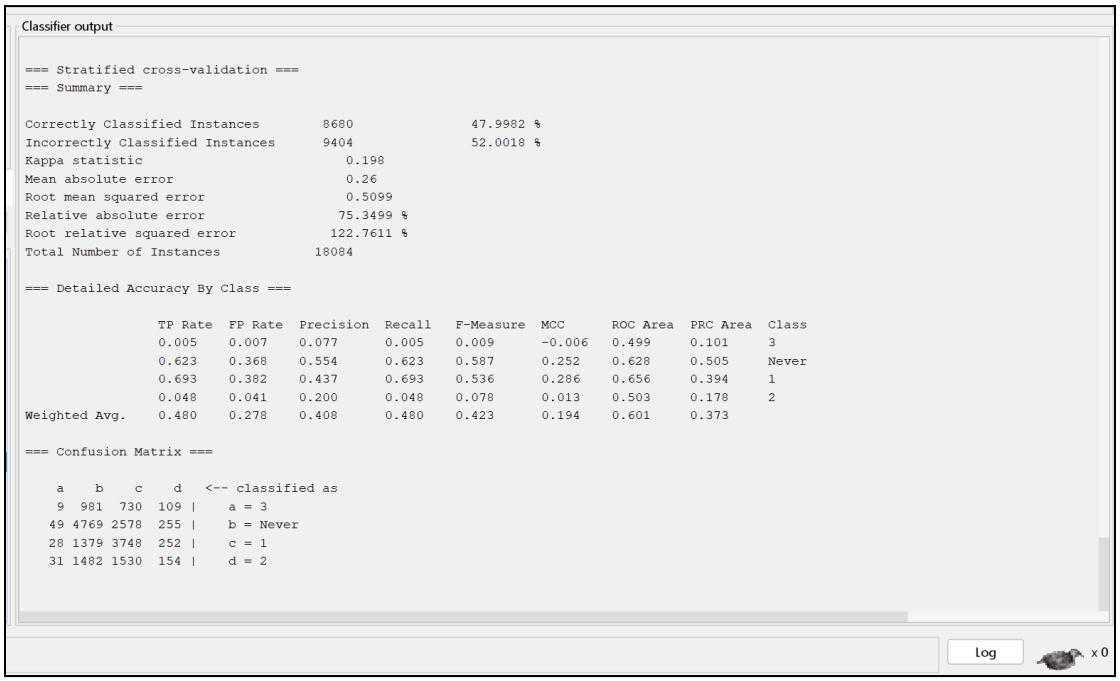
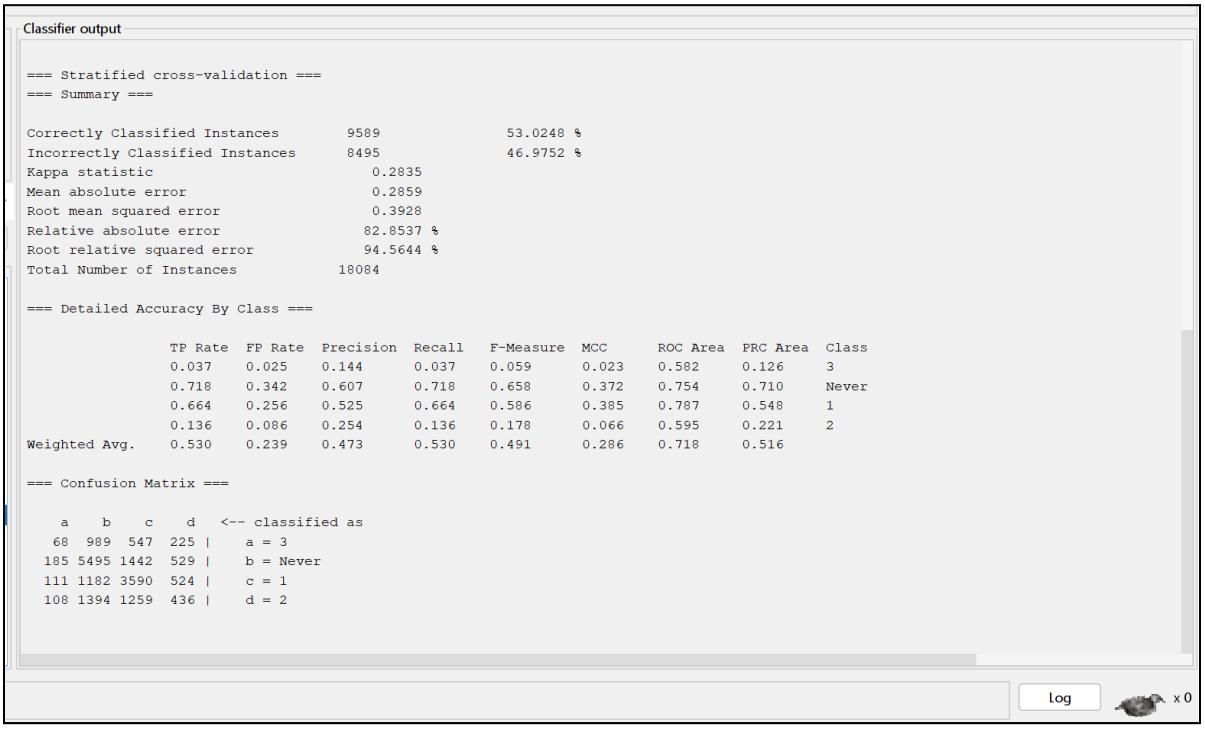


Figure 16: OneR OneRAttributeEval Correlation Result

RandomForest – OneRAttributeEval Attributes



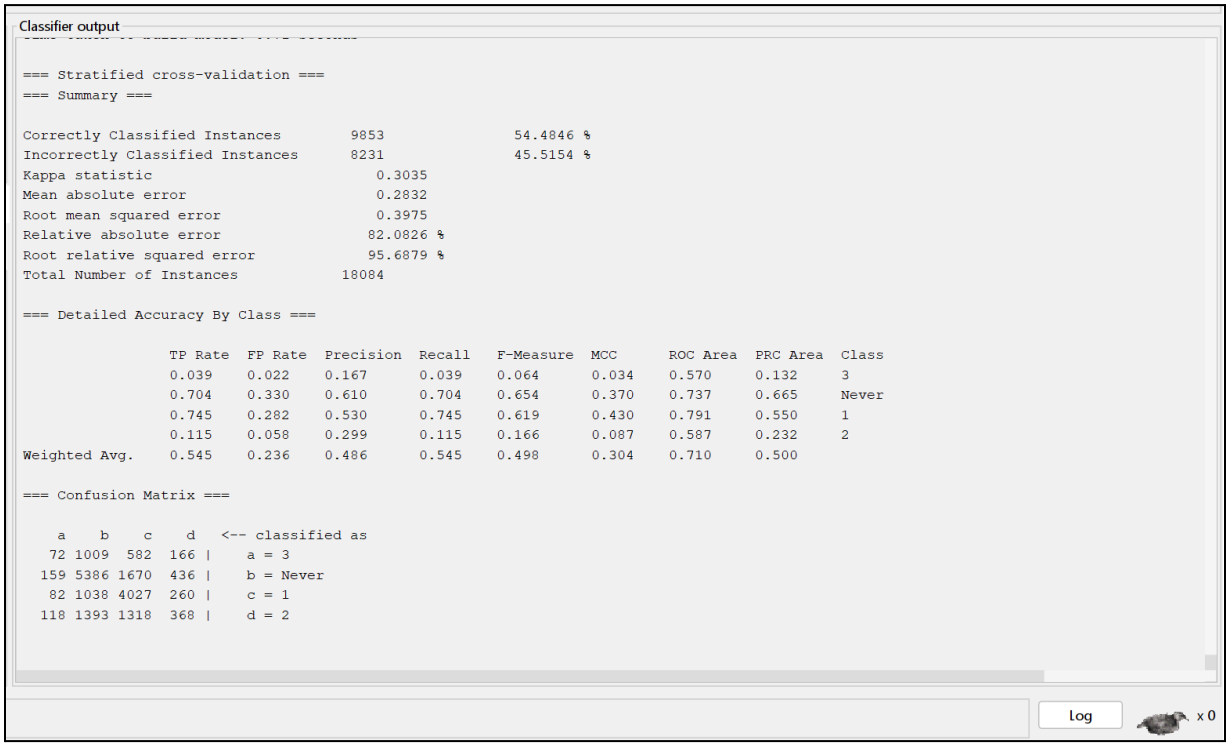
Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        9589               53.0248 %
Incorrectly Classified Instances      8495               46.9752 %
Kappa statistic                          0.2835
Mean absolute error                      0.2859
Root mean squared error                  0.3928
Relative absolute error                 82.8537 %
Root relative squared error             94.5644 %
Total Number of Instances            18084

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.037    0.025    0.144      0.037    0.059      0.023    0.582     0.126     3
               0.718    0.342    0.607      0.718    0.658      0.372    0.754     0.710     Never
               0.664    0.256    0.525      0.664    0.586      0.385    0.787     0.548     1
               0.136    0.086    0.254      0.136    0.178      0.066    0.595     0.221     2
Weighted Avg.  0.530    0.239    0.473      0.530    0.491      0.286    0.718     0.516

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
   68  989  547  225 |    a = 3
  185 5495 1442  529 |    b = Never
  111 1182 3590  524 |    c = 1
  108 1394 1259  436 |    d = 2
```

Log x 0

Figure 17: RandomForest OneRAttributeEval Correlation Result

J48 – WrapperSubsetEval Attributes



Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        9853               54.4846 %
Incorrectly Classified Instances      8231               45.5154 %
Kappa statistic                          0.3035
Mean absolute error                      0.2832
Root mean squared error                  0.3975
Relative absolute error                 82.0826 %
Root relative squared error             95.6879 %
Total Number of Instances            18084

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.039    0.022    0.167      0.039    0.064      0.034    0.570     0.132     3
               0.704    0.330    0.610      0.704    0.654      0.370    0.737     0.665     Never
               0.745    0.282    0.530      0.745    0.619      0.430    0.791     0.550     1
               0.115    0.058    0.299      0.115    0.166      0.087    0.587     0.232     2
Weighted Avg.  0.545    0.236    0.486      0.545    0.498      0.304    0.710     0.500

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
   72 1009  582  166 |    a = 3
  159 5386 1670  436 |    b = Never
   82 1038 4027  260 |    c = 1
  118 1393 1318  368 |    d = 2
```

Log x 0

Figure 18: J48 WrapperSubsetEval Correlation Result

Naive Bayes – WrapperSubsetEval Attributes

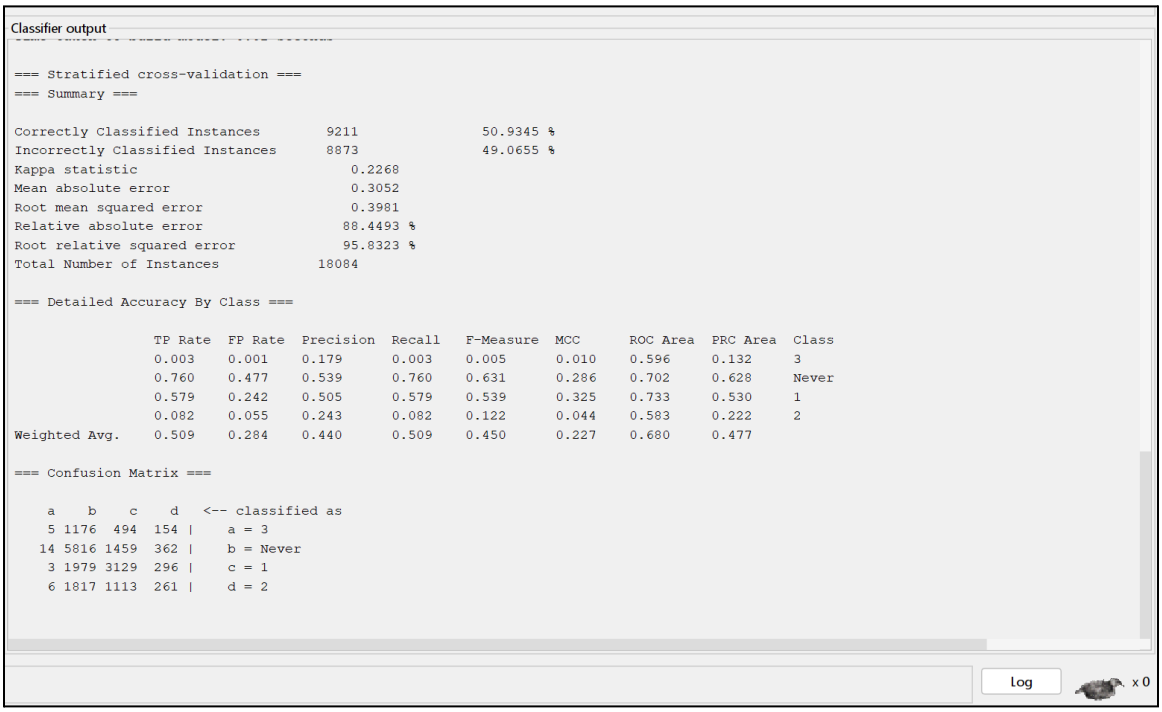


Figure 19: NaiveBayes WrapperSubsetEval Correlation Result

OneR – WrapperSubsetEval Attributes

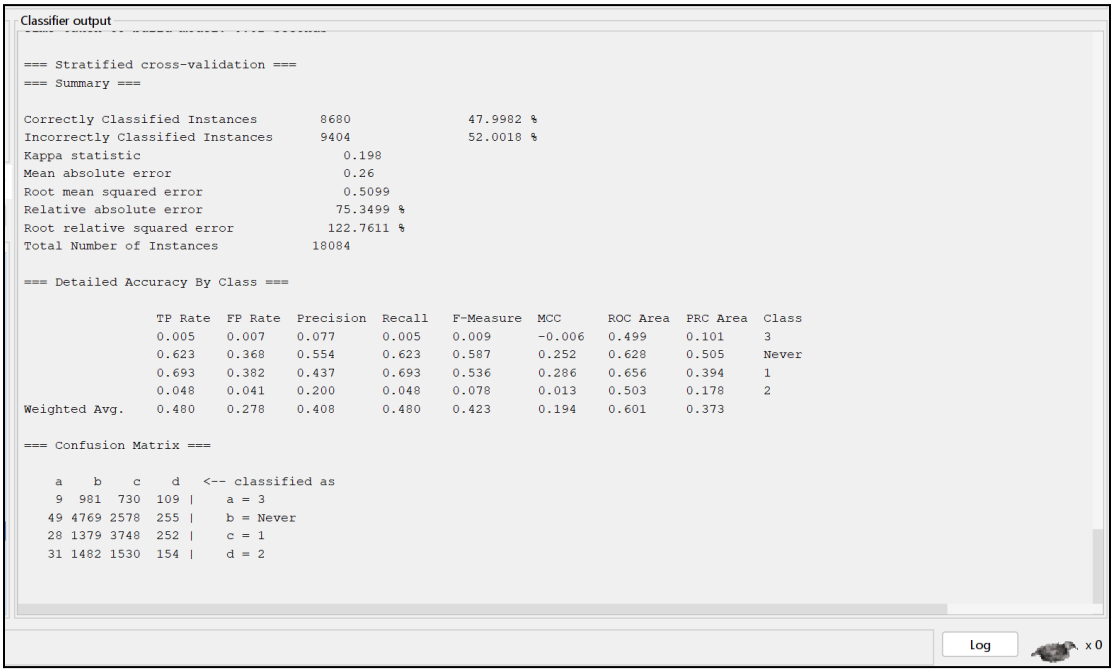


Figure 20: OneR WrapperSubsetEval Correlation Result

RandomForest – WrapperSubsetEval Attributes

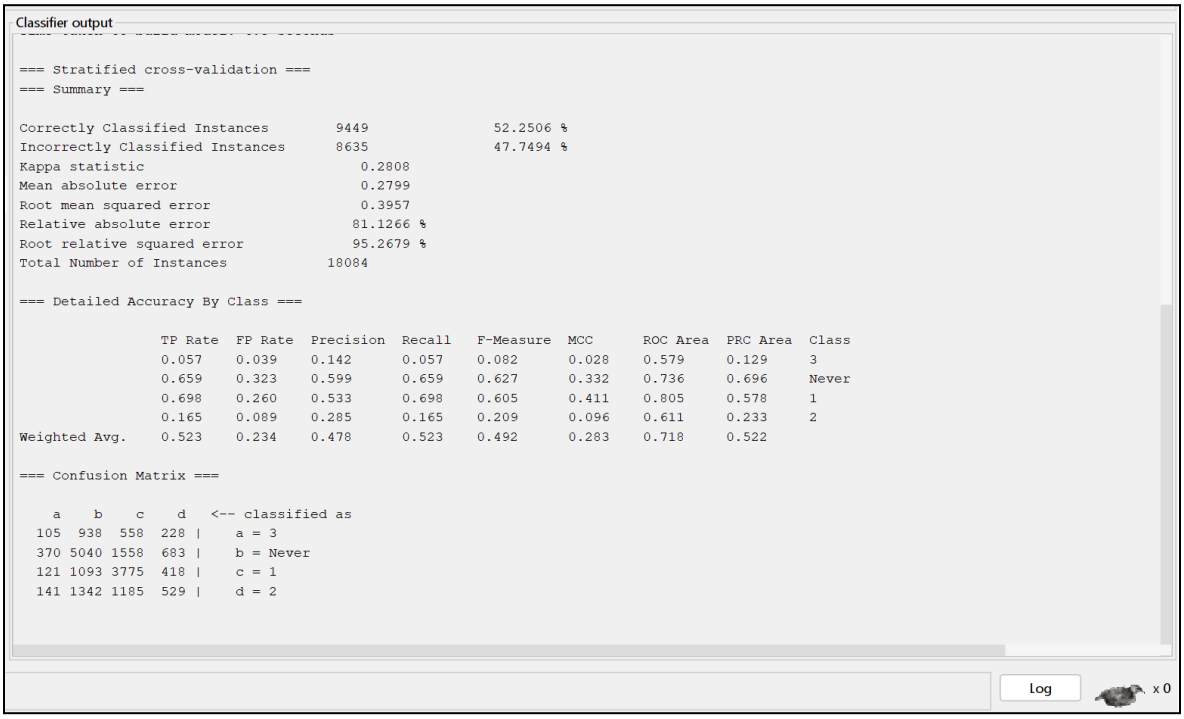


Figure 21: RandomForest WrapperSubsetEval Correlation Result

J48 – Self Selected Attributes

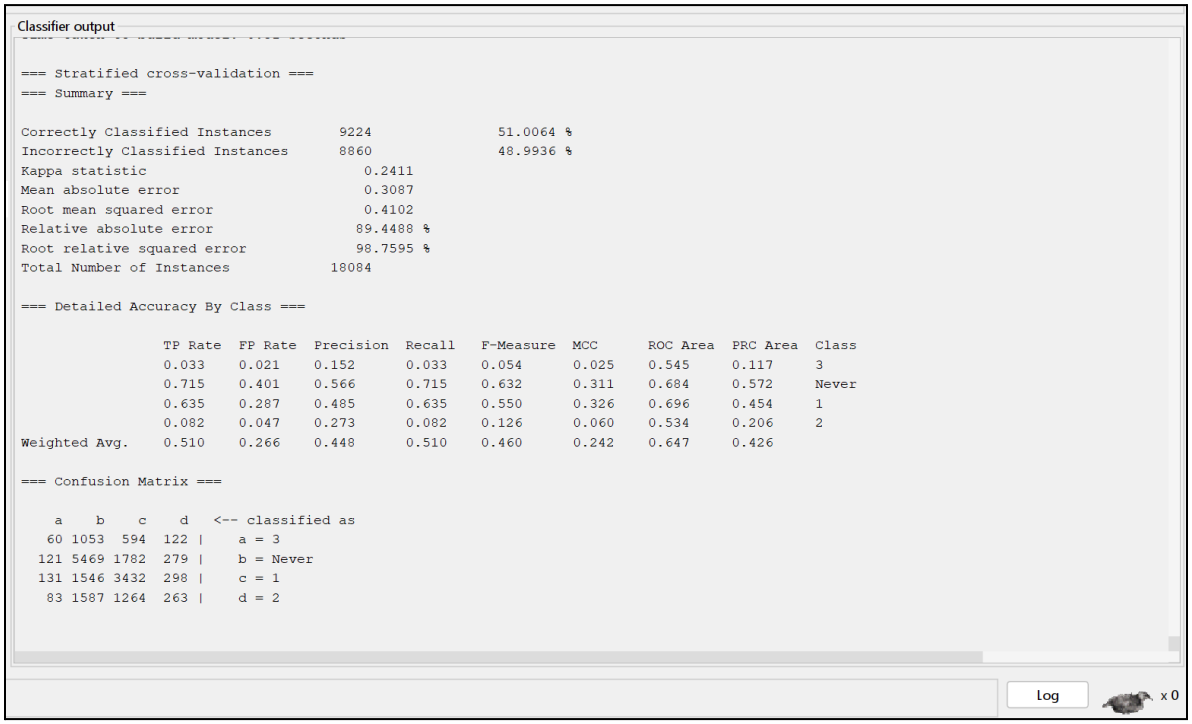


Figure 22: J48 Self Selection Correlation Result

Naive Bayes – Self Selected Attributes

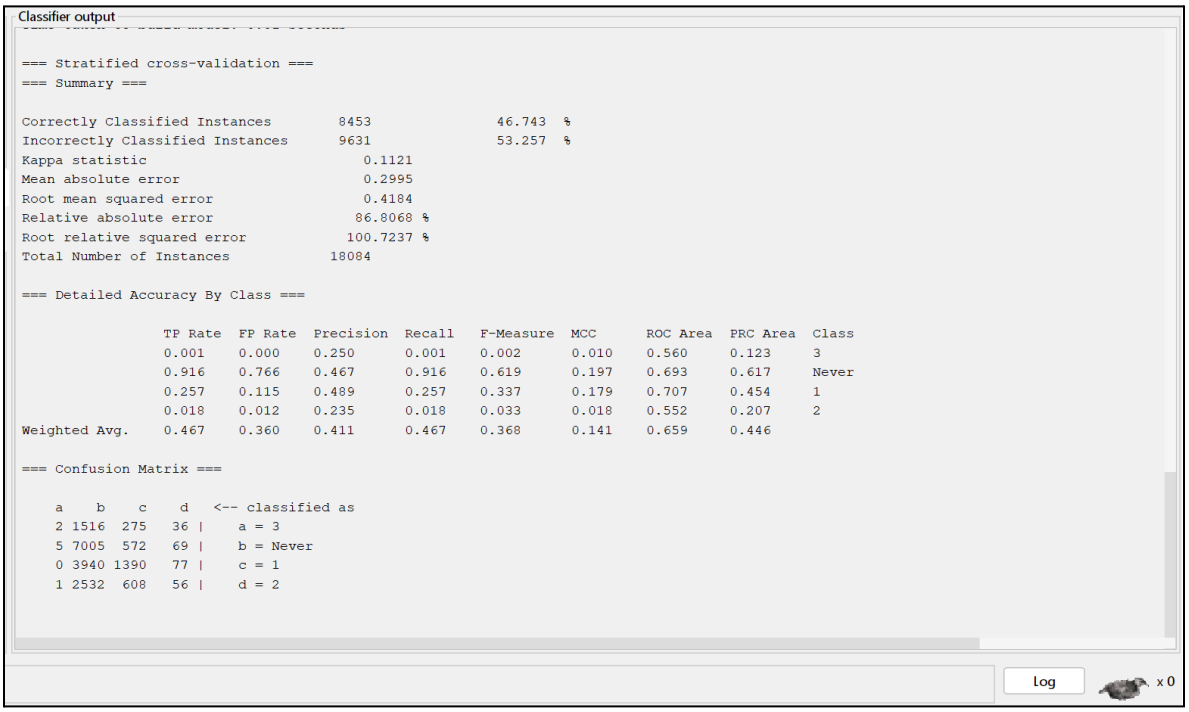


Figure 23: NaiveBayes Self Selection Correlation Result

OneR – Self Selected Attributes

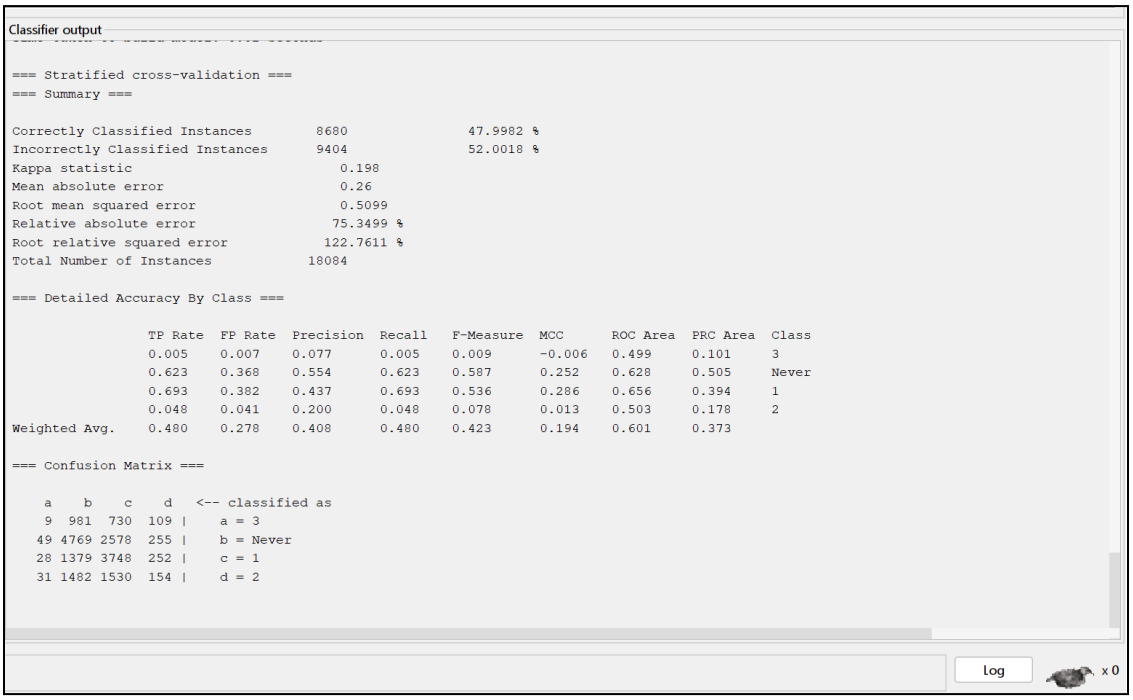


Figure 24: OneR Self Selection Correlation Result

Random Forest – Self Selected Attributes

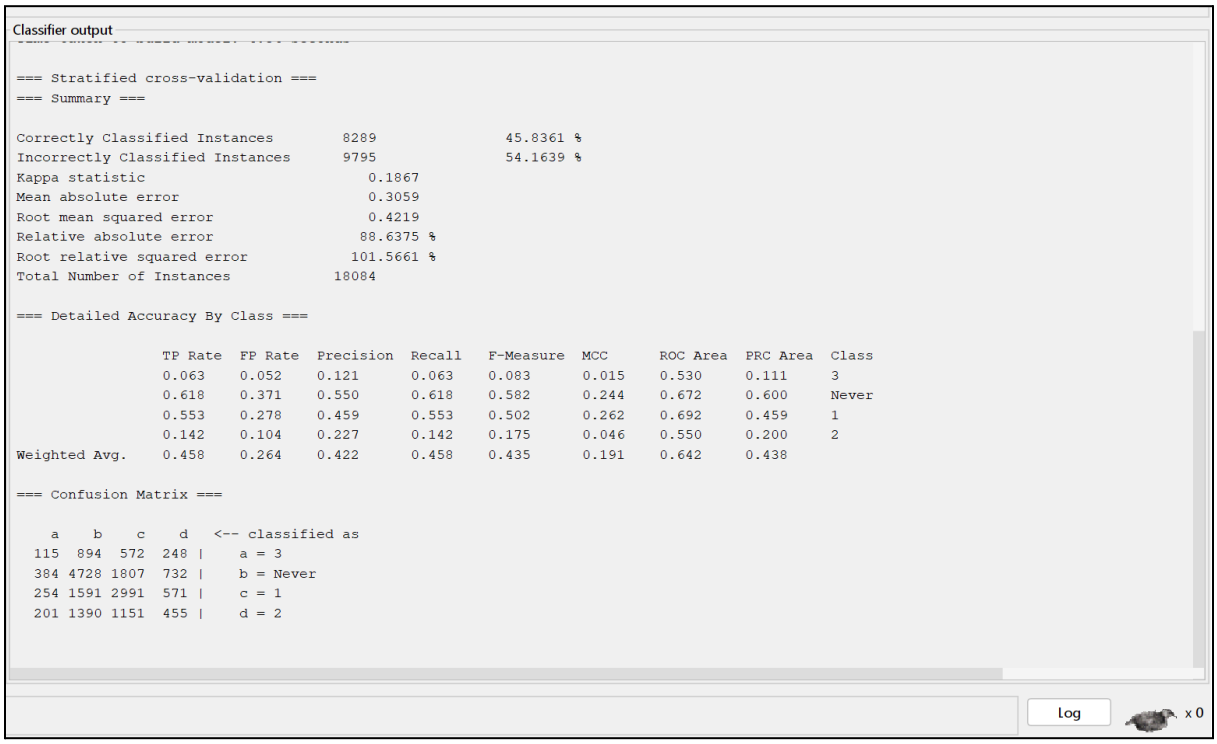


Figure 25: RandomForest Self Selection Correlation Result

Part 5 – Analysis & Conclusion

Part 5.1 - Analysis

Having built and tested our models, we then looked at their performance metrics to determine which model performed best. In the context of our dataset, a false positive denotes a criminal that is predicted to commit a second crime but will not, and a false negative denotes a criminal that is not predicted to commit a second crime but does. Because a false negative would be more detrimental or "costly," we decided to look at recall in addition to accuracy. This is because recall measures the proportion of true positive values to the total number of actually positive instances.

Below are five histograms, one for each set of attributes, that compare the accuracy of the four classification algorithms:

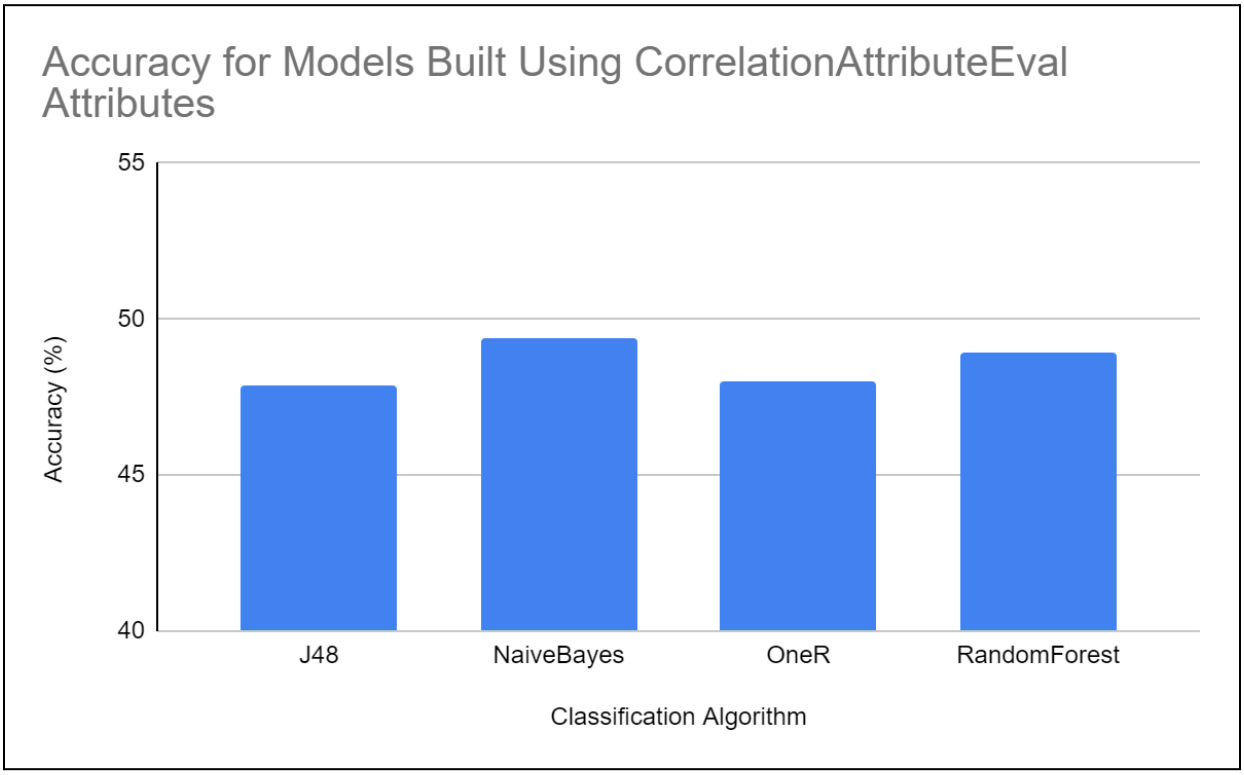


Figure 26: Accuracy for Models Using CorrelationAttributeEval

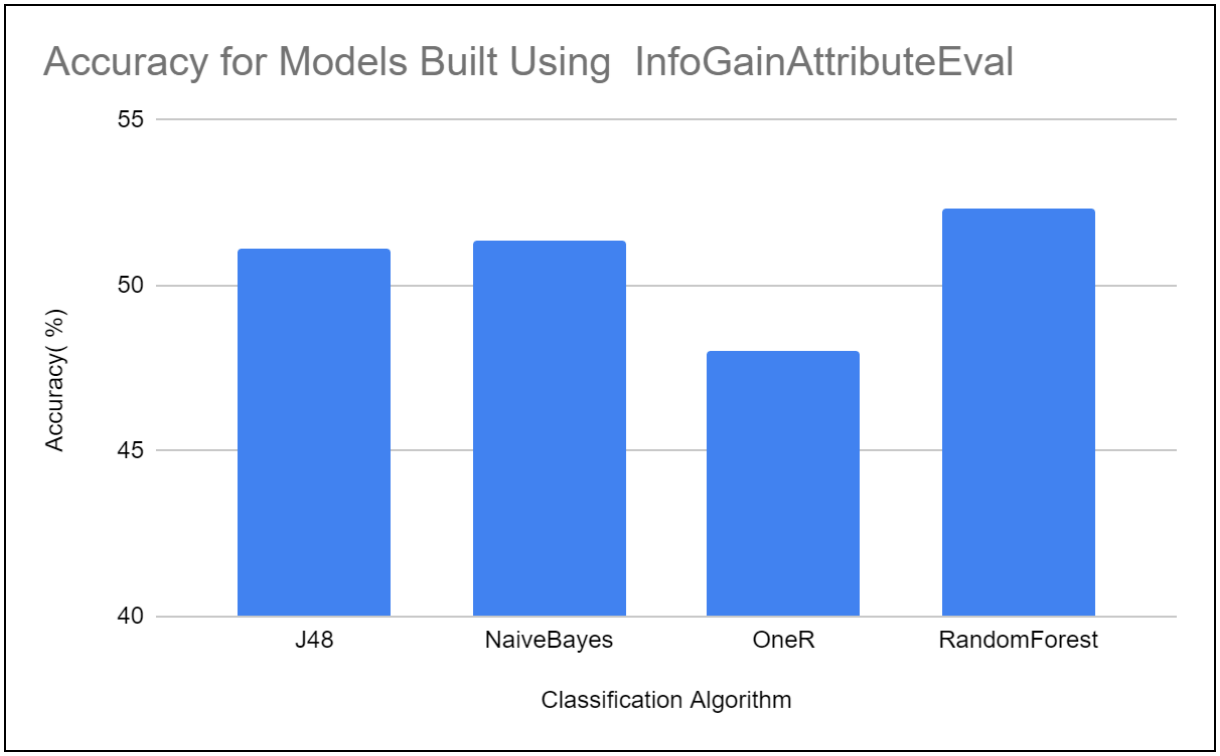


Figure 27: Accuracy for Models Using InfoGainAttributeEval

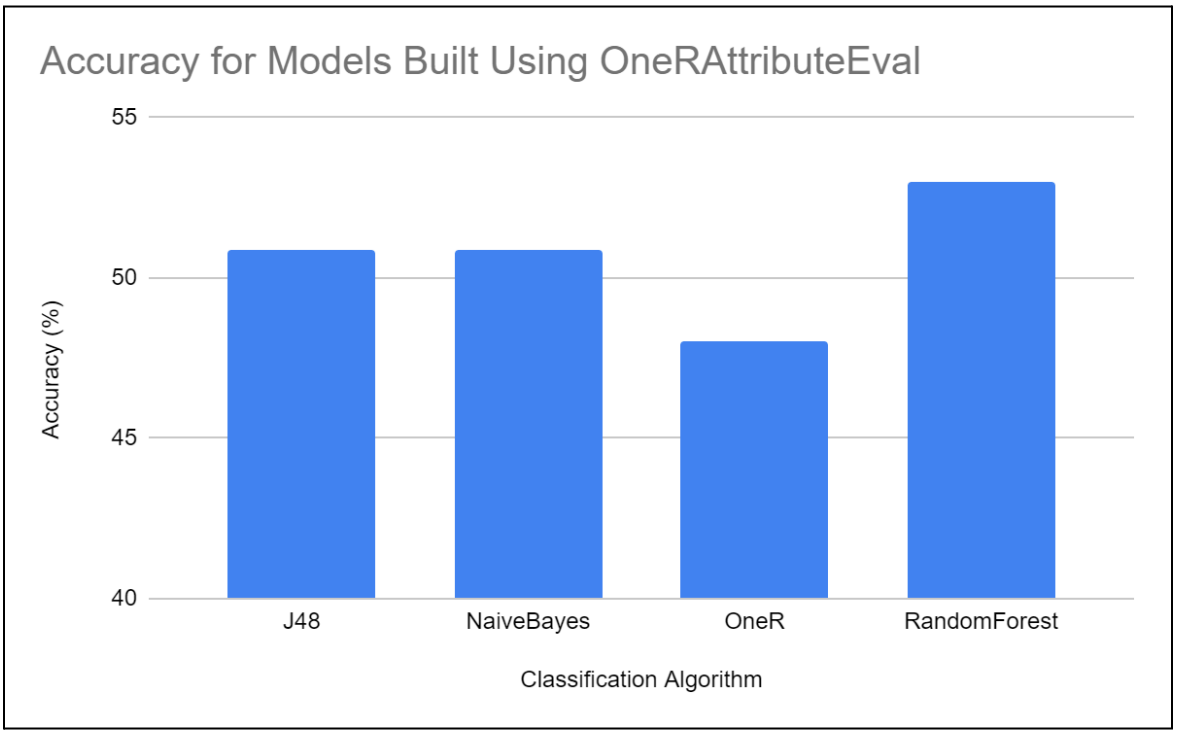


Figure 28: Accuracy for Models Using OneRAttributeEval

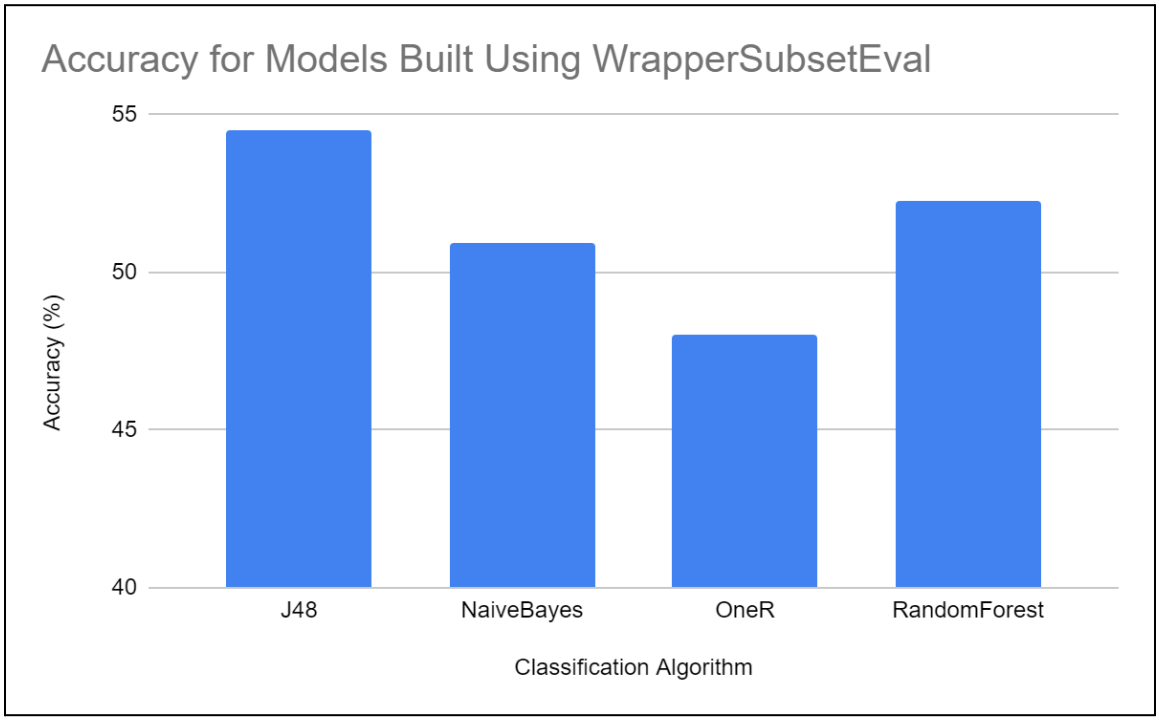


Figure 29: Accuracy for Models Using WrapperSubsetEval

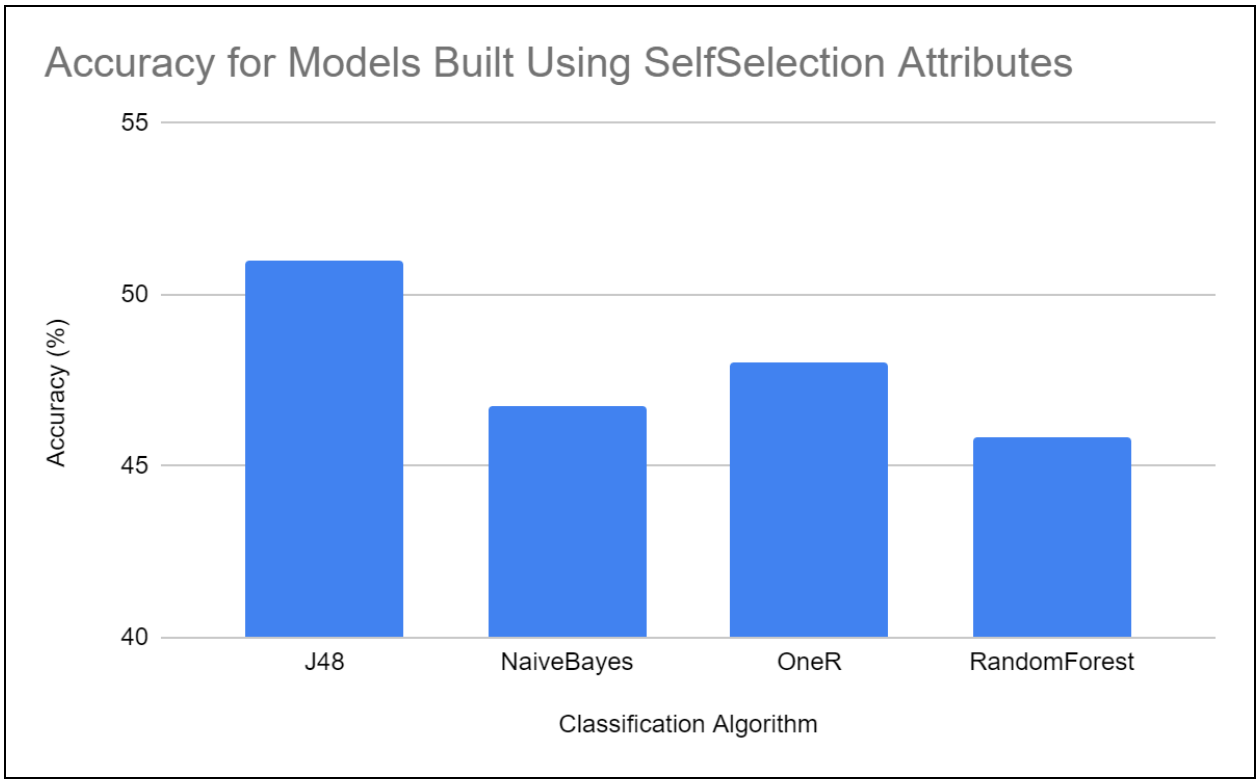


Figure 30: Accuracy for Models Using Self-Selection

In general, models built using attributes selected by CorrelationAttributeEval and SelfSelection had lower accuracy than models built using InfoGainEval, OneRAttributeEval, and WrapperSubsetEval. The model with the highest accuracy utilized attributes selected by the WrapperSubsetEval attribute selector, and used a J48 classification algorithm. This model had an accuracy of 54.4846.

Similarly, 5 histograms are depicted in the following *Figures*. These histograms depict the recall for each model. We calculated recall using Macroaveraging due to the class imbalance in our dataset.

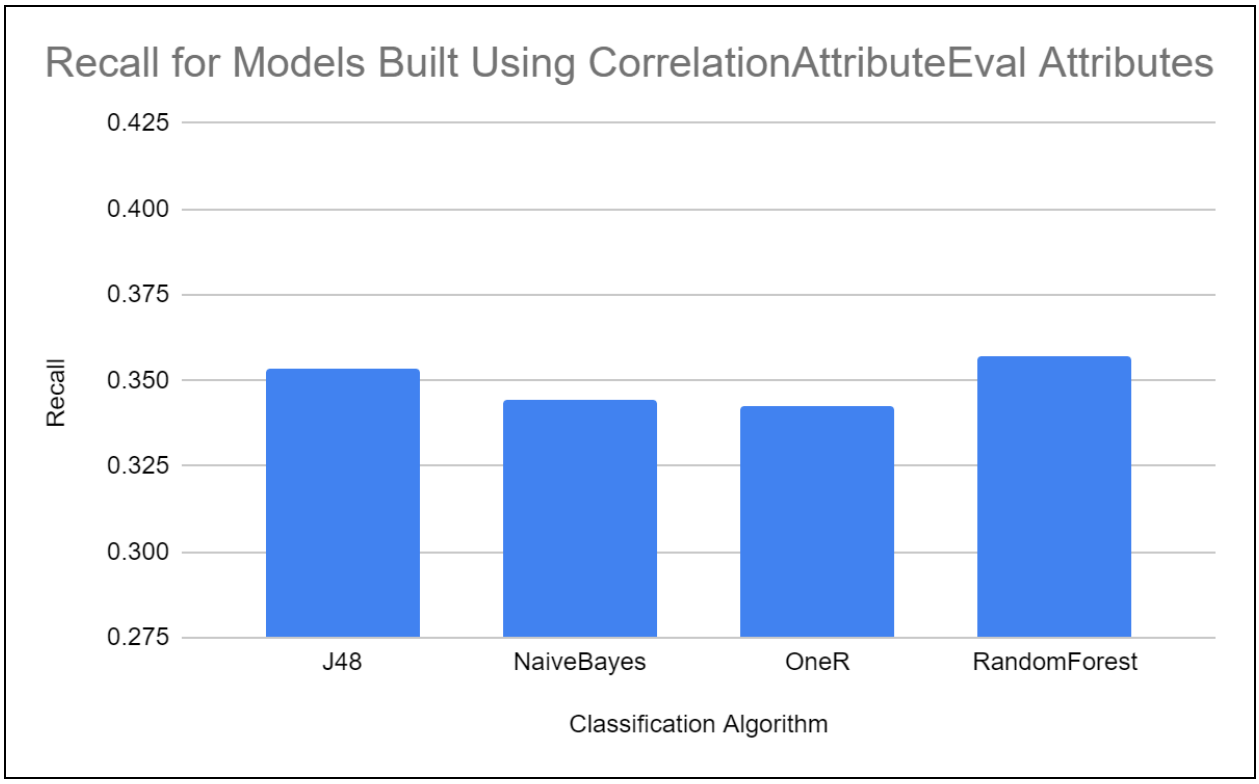


Figure 31: Recall for Models Using CorrelationAttributeEval

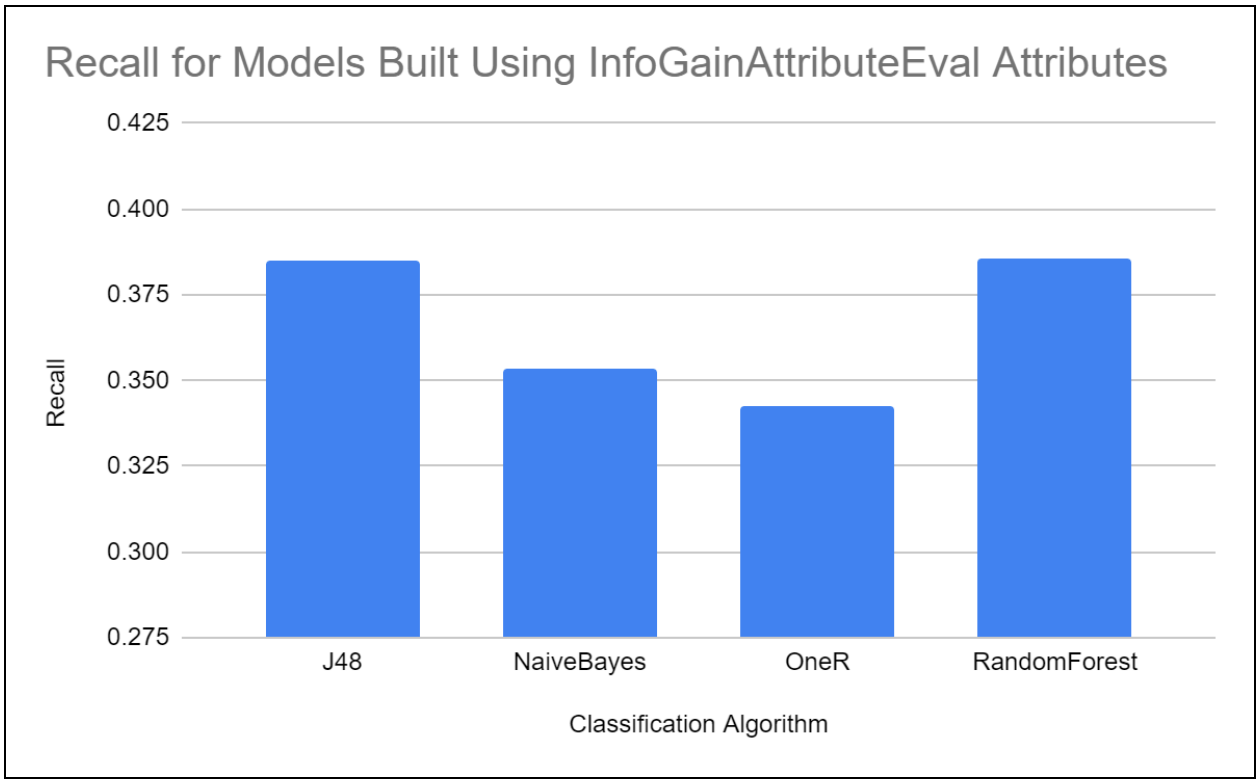


Figure 32: Recall for Models Using InfoGainAttributeEval

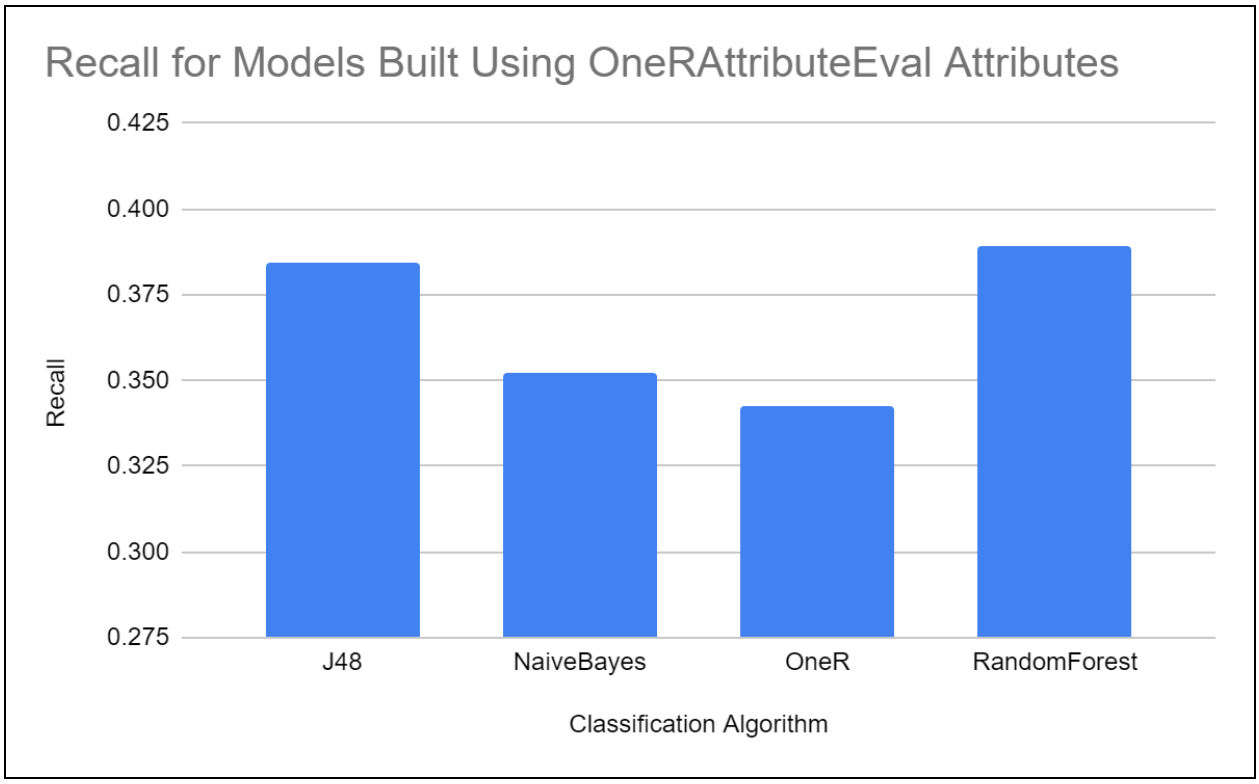


Figure 33: Recall for Models Using OneRAttributeEval

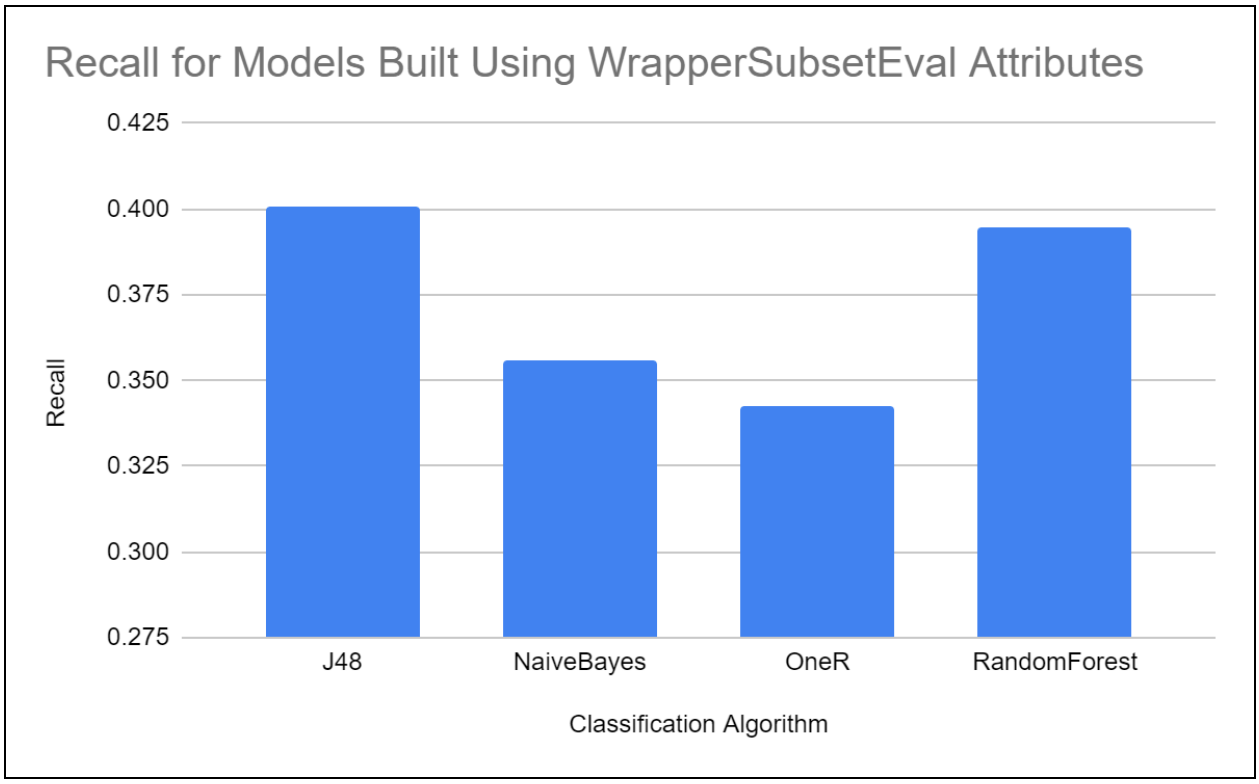


Figure 34: Recall for Models Using WrapperSubsetEval

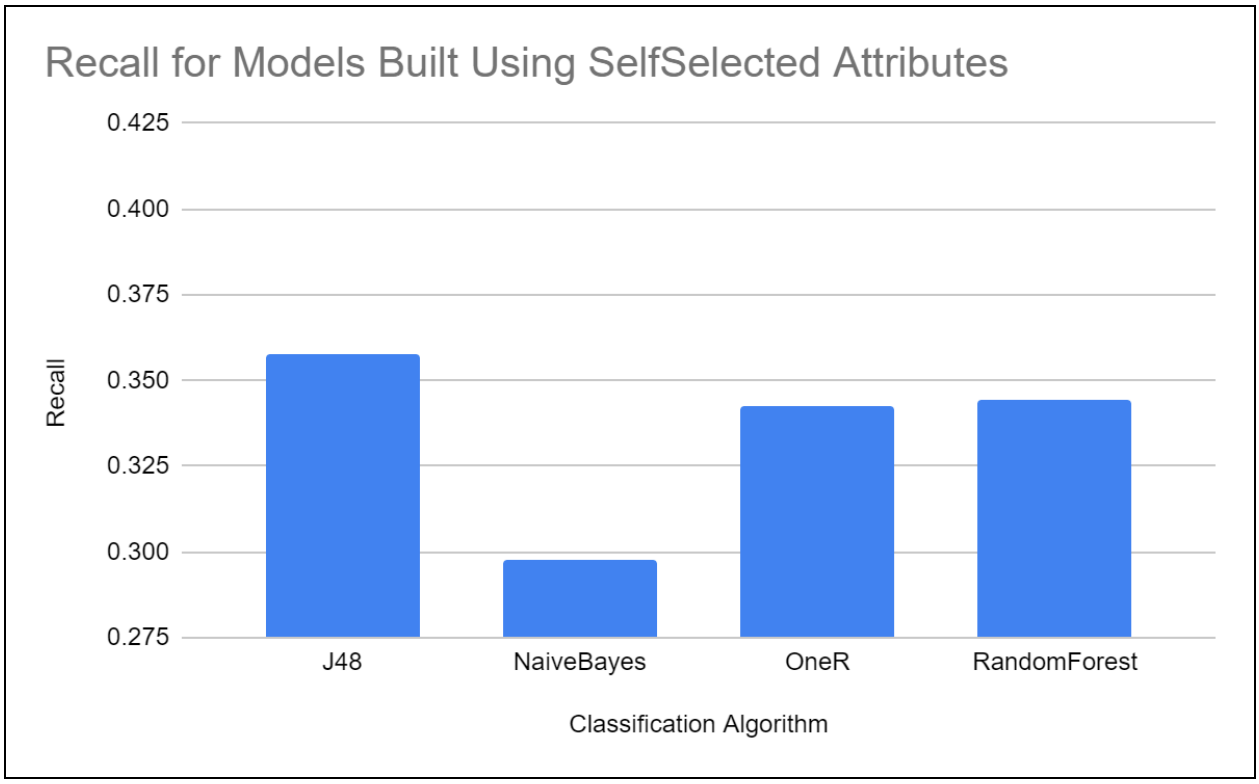


Figure 35: Recall for Models Using Self Selected Attributes

Once more, models built using attributes selected by CorrelationAttributeEval and SelfSelection have lower recalls than models built using InfoGainEval, OneRAttributeEval, and WrapperSubsetEval. The model with the highest recall, once again, utilized attributes selected by the WrapperSubsetEval attribute selector and used a J48 classification algorithm. This model had a recall of 0.400802311309.

Because the model built utilizing attributes selected by the WrapperSubsetEval attribute selector and a J48 classification algorithm had both the highest accuracy and the highest recall, this model is the best model for predicting recidivism.

Part 5.2 – Discussion of Error and Conclusion

While we identified some potentially pertinent factors in predicting the likelihood of recidivism, none of our models achieved an accuracy higher than 55%. One potential reason for this is that we separated out criminals who committed crimes after release from prison into three categories: within 1 year, 2 years, and 3 years. However, criminals that commit a second crime after release from jail likely share similar characteristics, making it difficult for our model to distinguish between these cases. In order to test this prediction, we removed all instances with the value "2" years or "3" years from the dataset. We then used our previous best model (WrapperSubsetEval Attributes with J48) to predict whether a criminal never committed a second crime, or committed a crime within one year. This model achieved an accuracy of 77.2% as seen below:

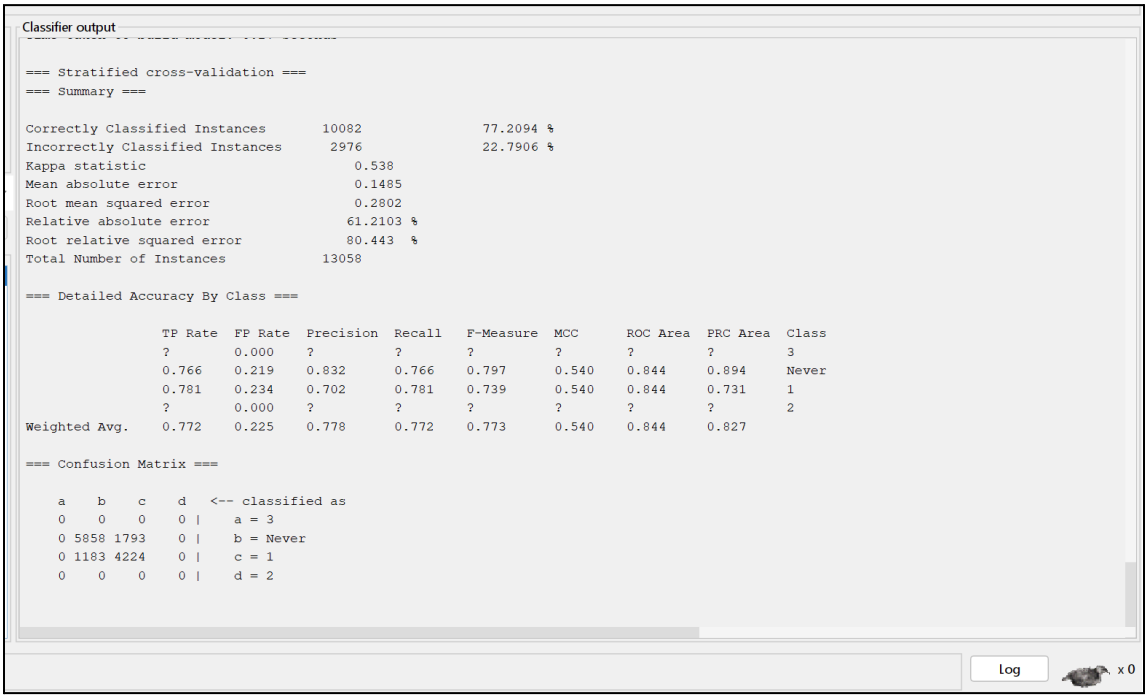


Figure 36: Accuracy of Classification Model for Instances Labeled "1" or "Never"

However, removing the label “Never” and running the model produced an accuracy of 59.8%, as seen below:

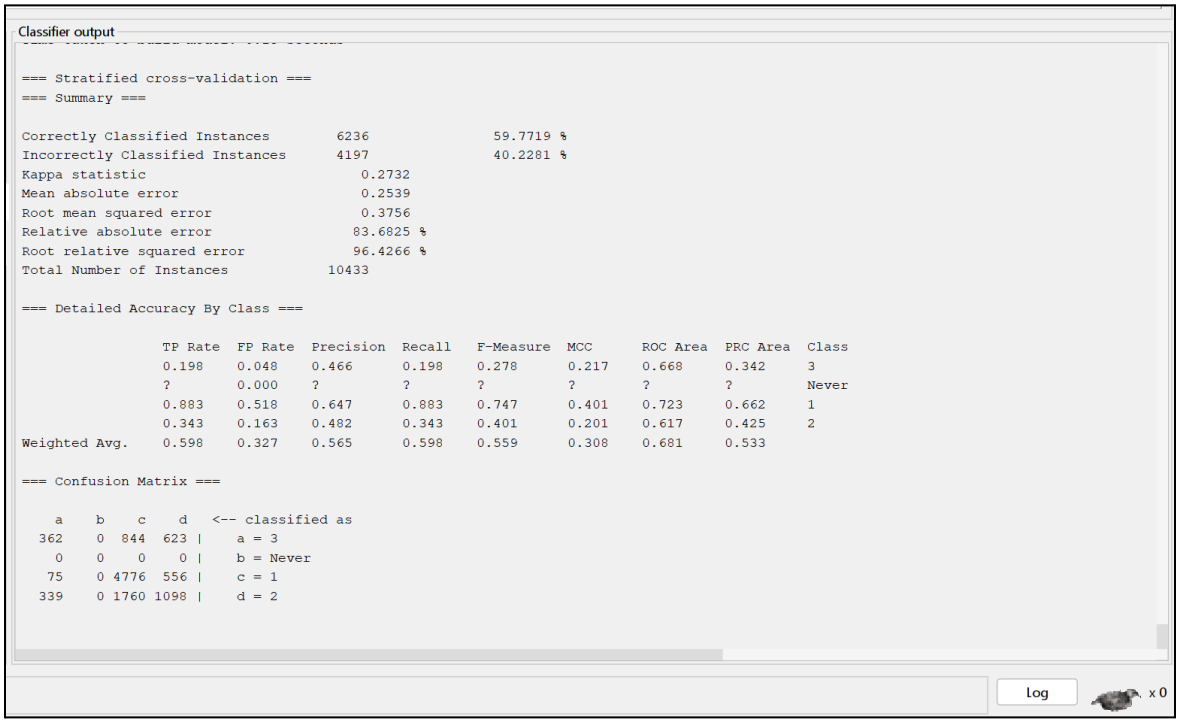


Figure 37: Accuracy of Classification Model for Instances Labeled “1”, “2”, or “3”

This was only slightly better than our best model with all labels. Like we predicted, these results potentially indicate that the model has a difficult time distinguishing between criminals that commit crimes 1, 2, or 3 years after release from jail.

Another potential source of error is the class imbalances in our dataset. Because so few criminals committed crimes within 2 or 3 years of release from prison, our models may have been unable to predict these class labels very well. Analyzing the correlation matrices of our 20 original models, we saw that models tended to inaccurately predict “Never” or within “1” year very frequently, while almost never predicting “2” or “3”. This likely indicates that our class

imbalances (specifically, the abundance of the class label“Never” and within “1” year), were causing the model to predict “Never” and “1” most of the time.

While this risk assessment model would prove highly useful in a real world context, with such a low accuracy, more analysis is required to build a model accurate enough to be worth relying upon.