**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 3**](#_30j0zll)

[**Part 2 – Description of Dataset 3**](#_1fob9te)

[**Part 3 – Pre-Processing**](#_3znysh7) **9**

[**Part 4 – Attribute Selection Algorithms & Model Classifiers Used**](#_2et92p0) **16**

[**Part 5 – Results and Analysis**](#_tyjcwt) **29**

[**Part 6 – Conclusion 23**](#_3dy6vkm)

# **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*

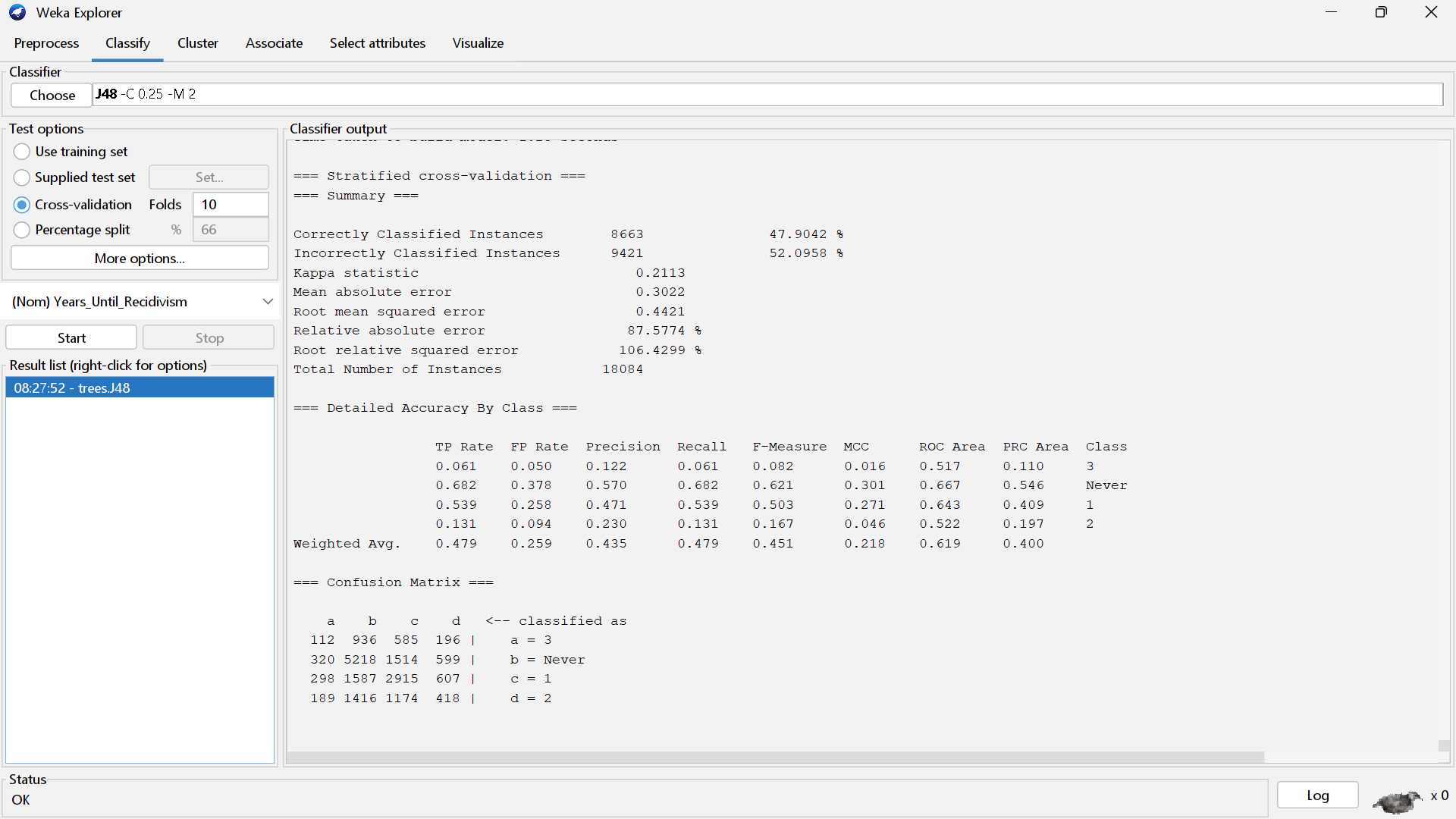
**Part 4.2 Classifier Models**

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

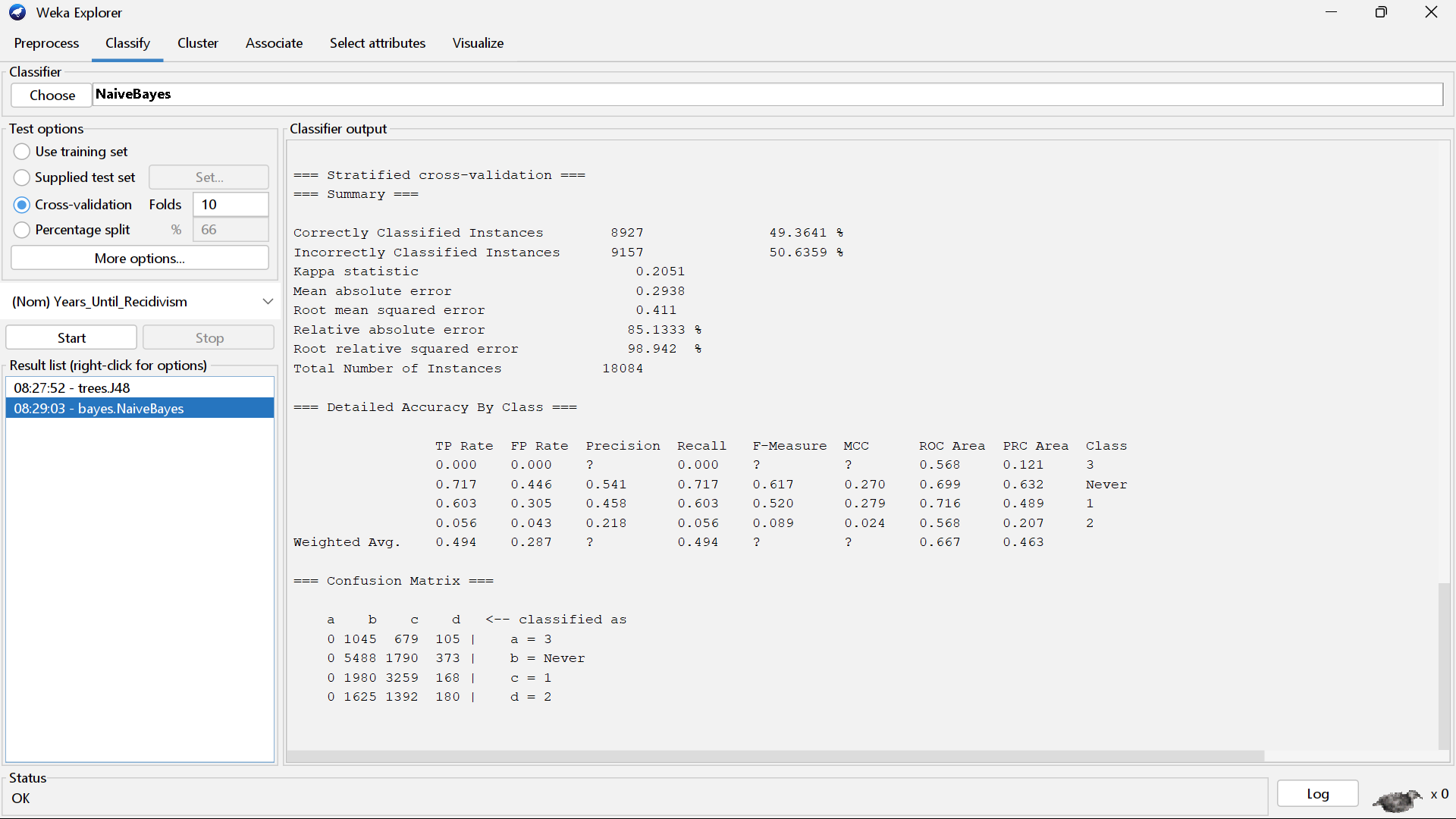
In WEKA, we created and tested all 20 models using K-Fold cross validation with 10 folds. The outputs for these models is shown below:

J48 – CorrelationAttributeEval Attributes



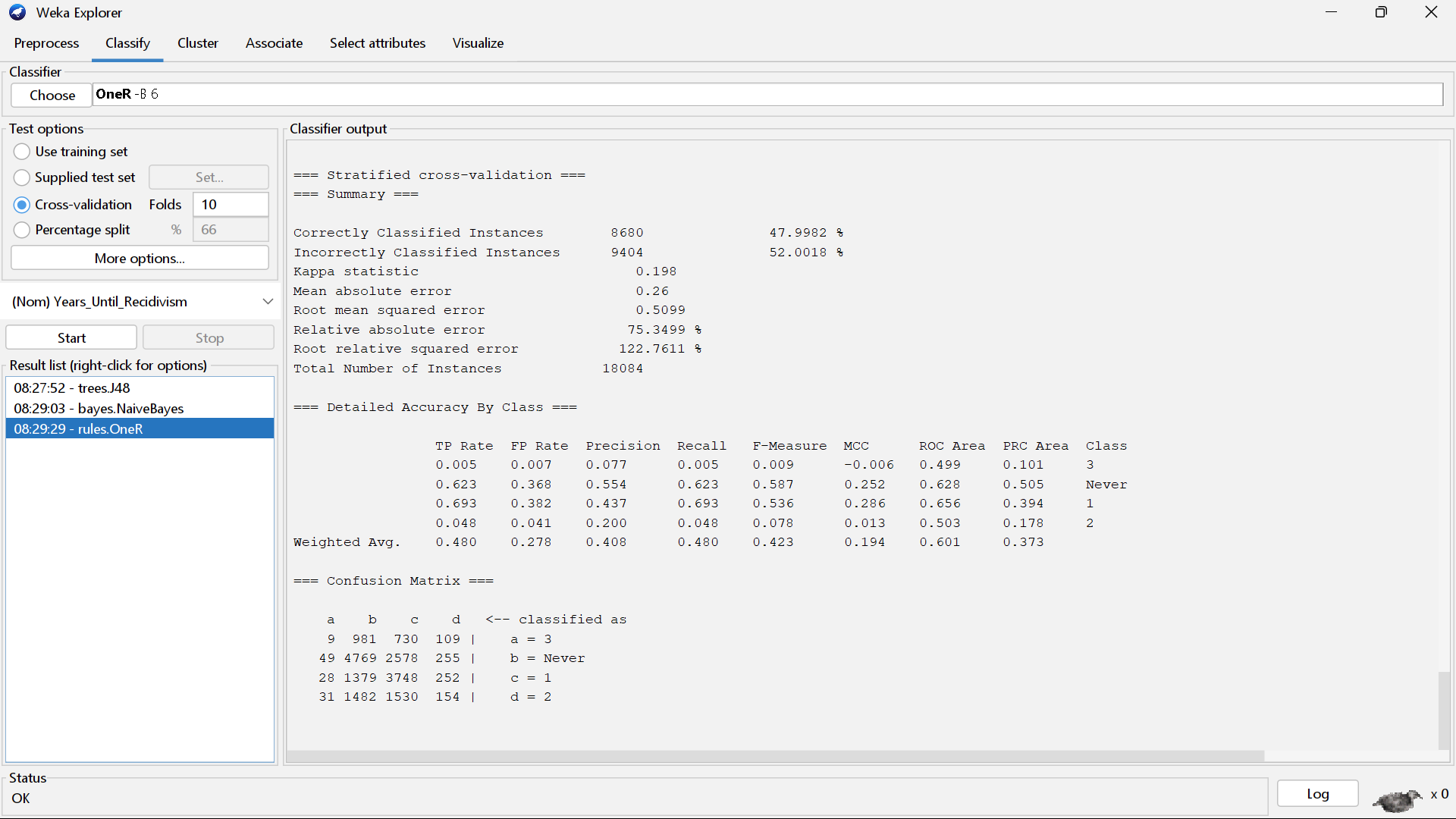
*Figure 7: J48 CorrelationAttributeEval Correlation Result*

Naive Bayes – CorrelationAttributeEval Attributes



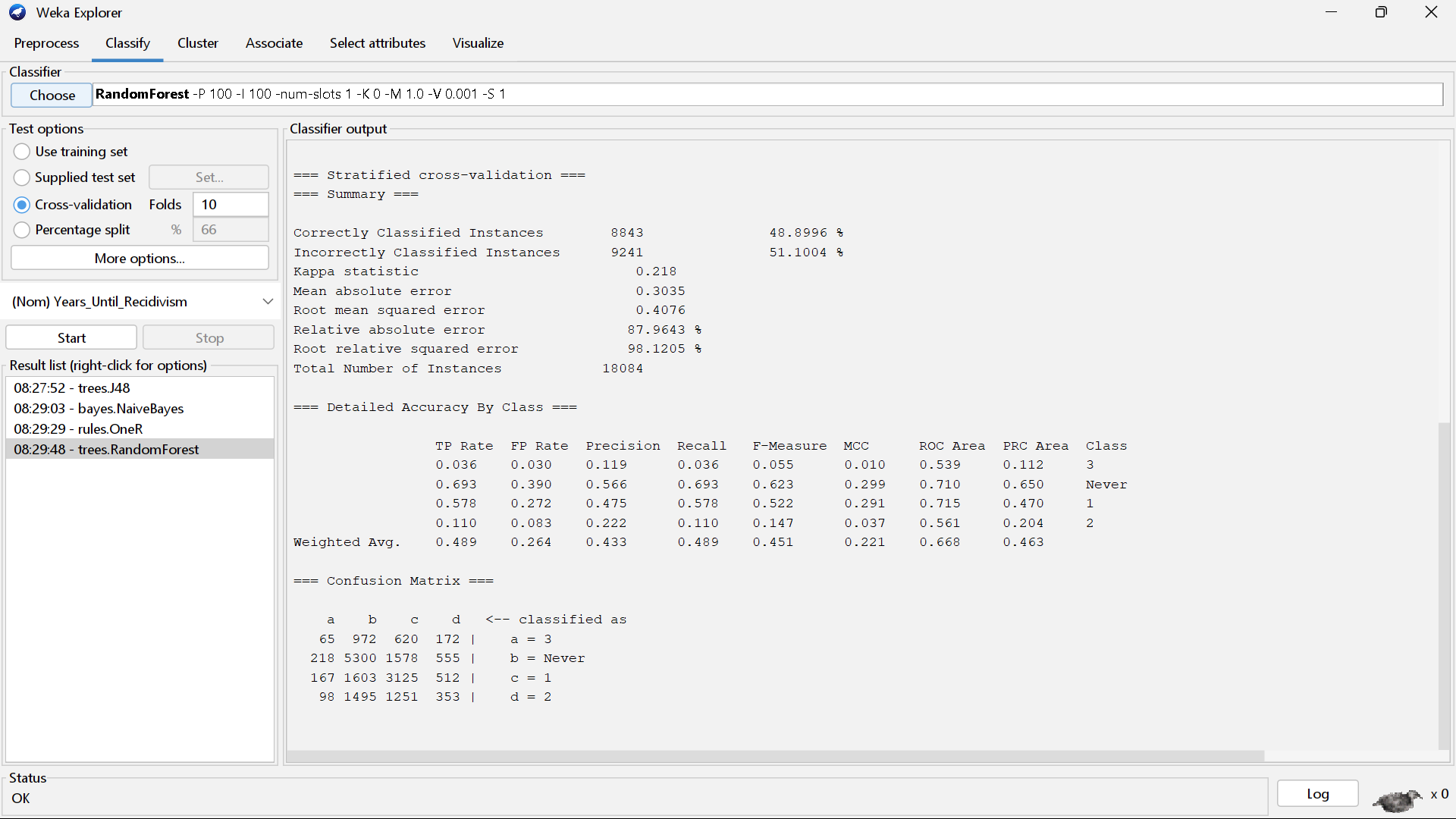
*Figure 8: NaiveBayes CorrelationAttributeEval Correlation Result*

OneR – CorrelationAttributeEval Attributes



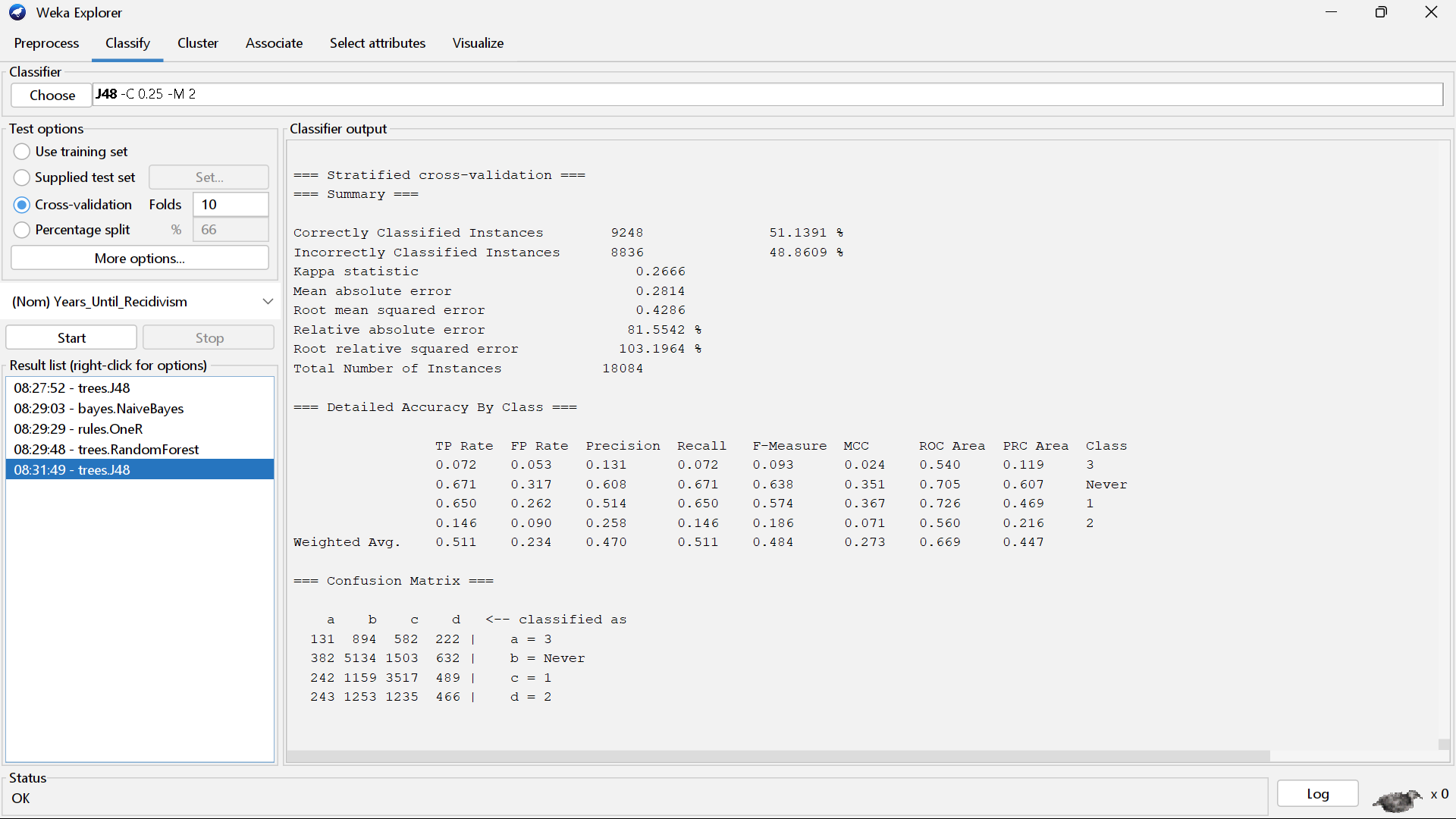
*Figure 9: OneR CorrelationAttributeEval Correlation Result*

RandomForest – CorrelationAttributeEval Attributes



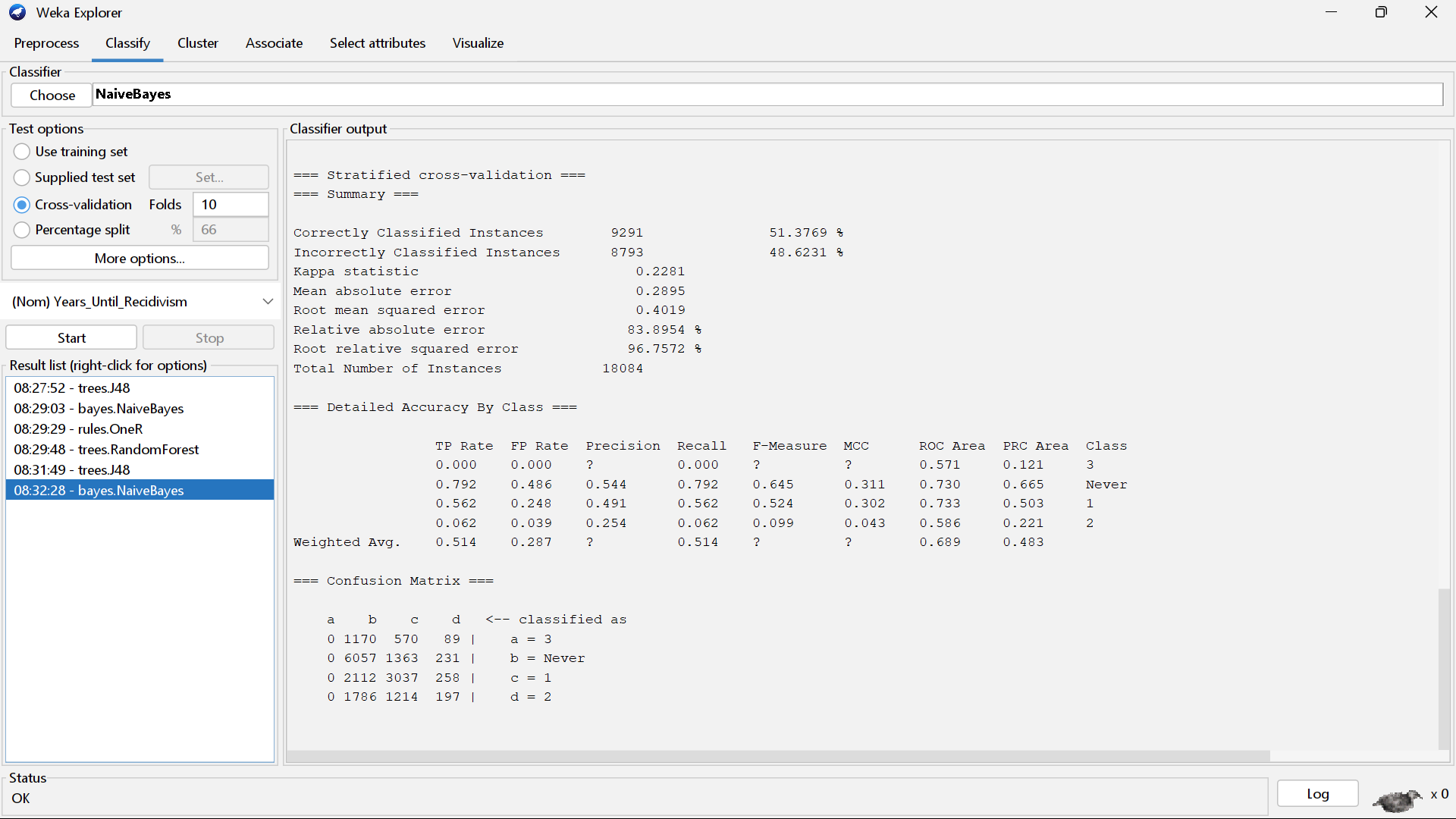
*Figure 10: RandomForest CorrelationAttributeEval Correlation Result*

J48 – InfoGainAttributeEval Attributes



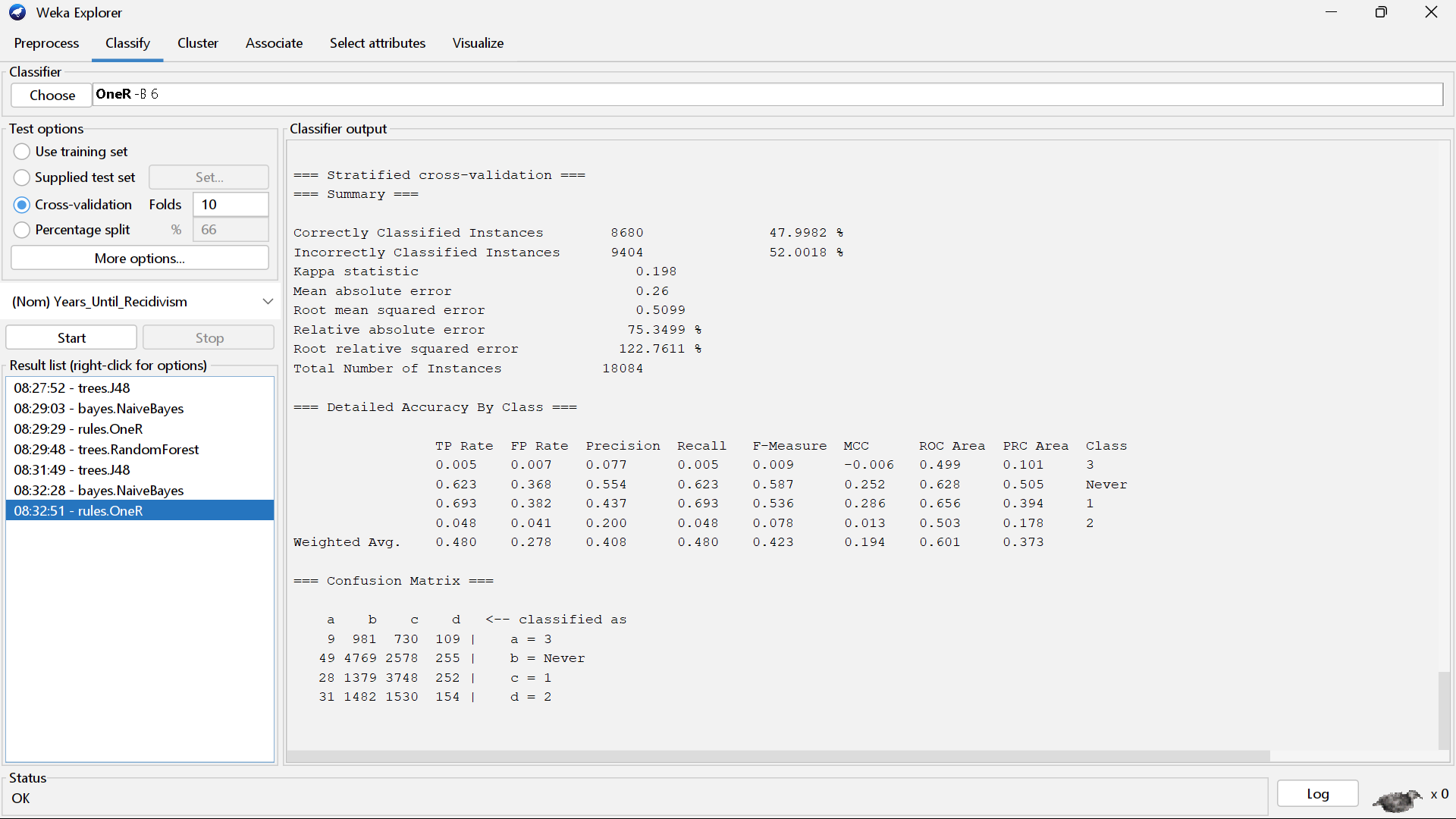
*Figure 11: J48 InfoGainAttributeEval Correlation Result*

Naive Bayes – InfoGainAttributeEval Attributes



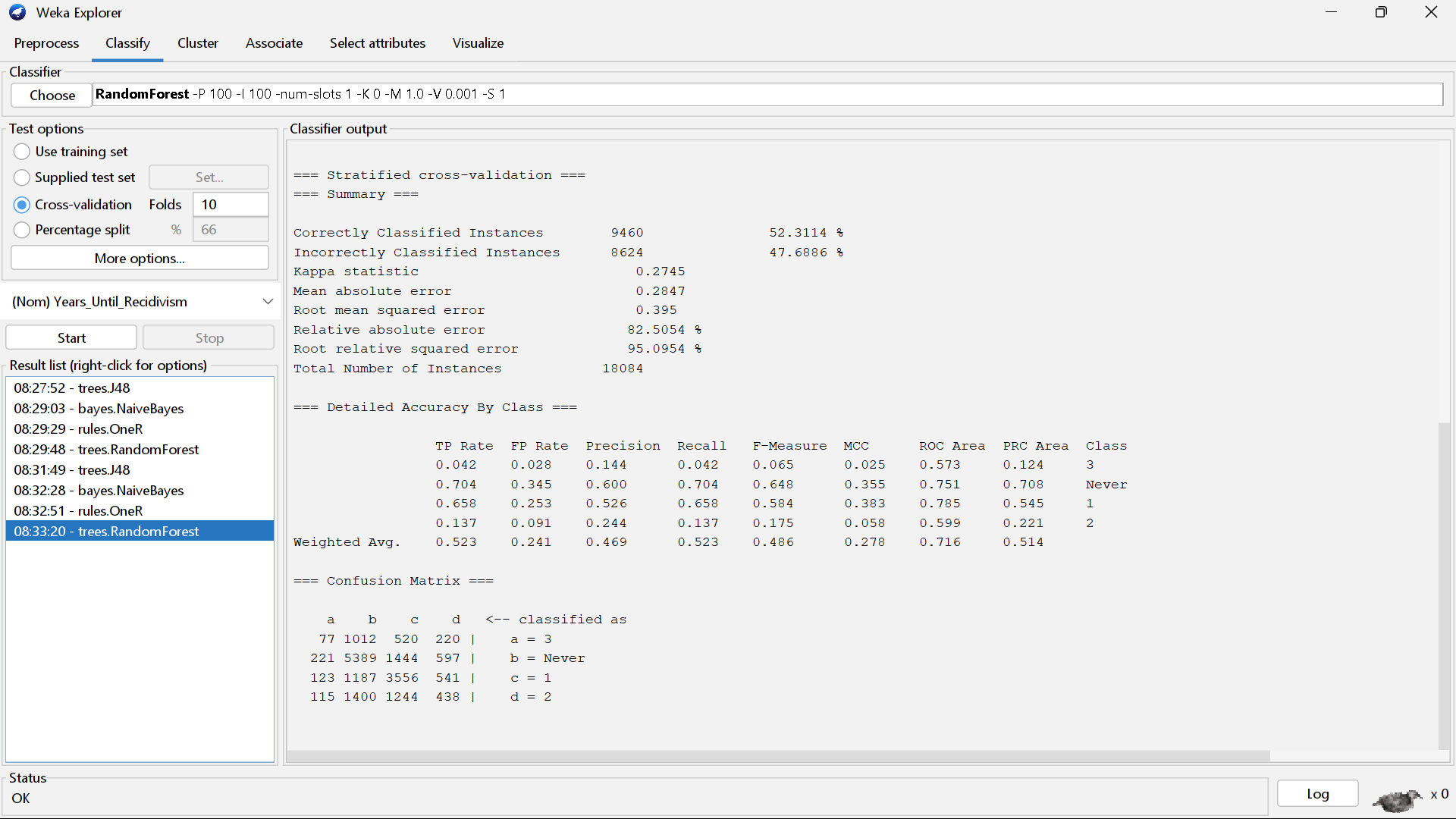
*Figure 12: NaiveBayes InfoGainAttributeEval Correlation Result*

One R – InfoGainAttributeEval Attributes



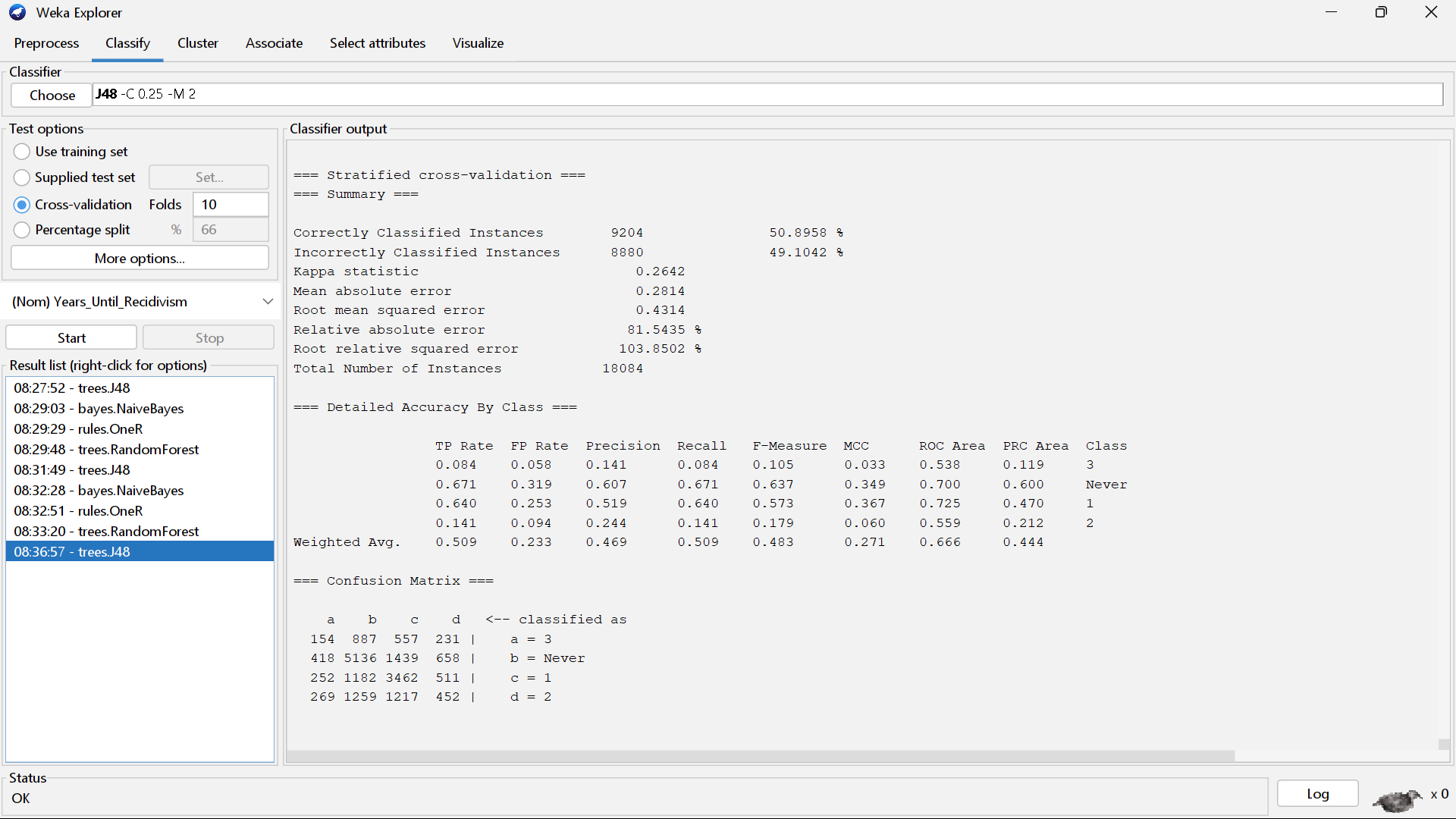
*Figure 12: OneR InfoGainAttributeEval Correlation Result*

RandomForest – InfoGainAttributeEval Attributes



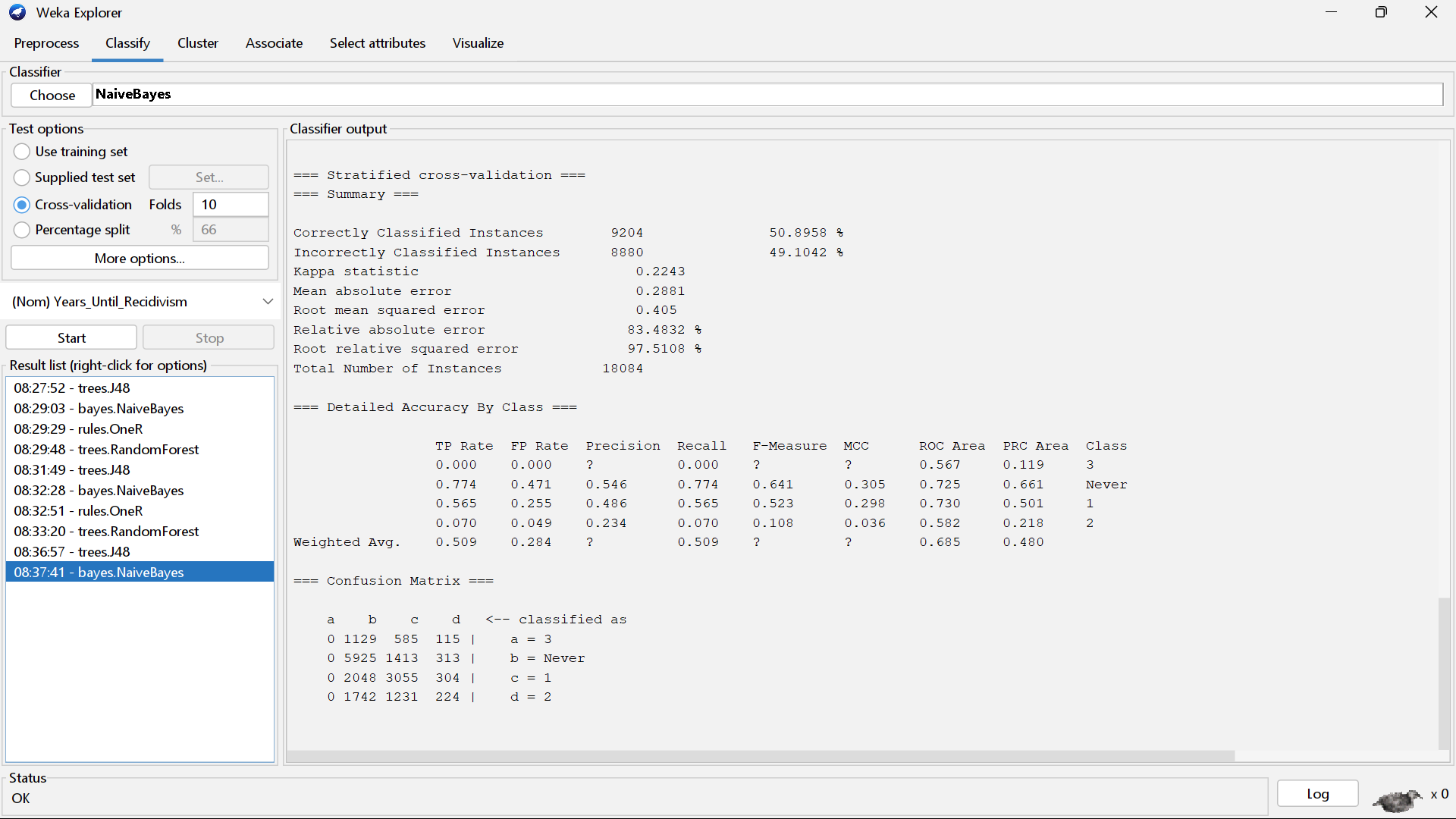
*Figure 13: RandomForest InfoGainAttributeEval Correlation Result*

J48 – OneRAttributeEval Attributes



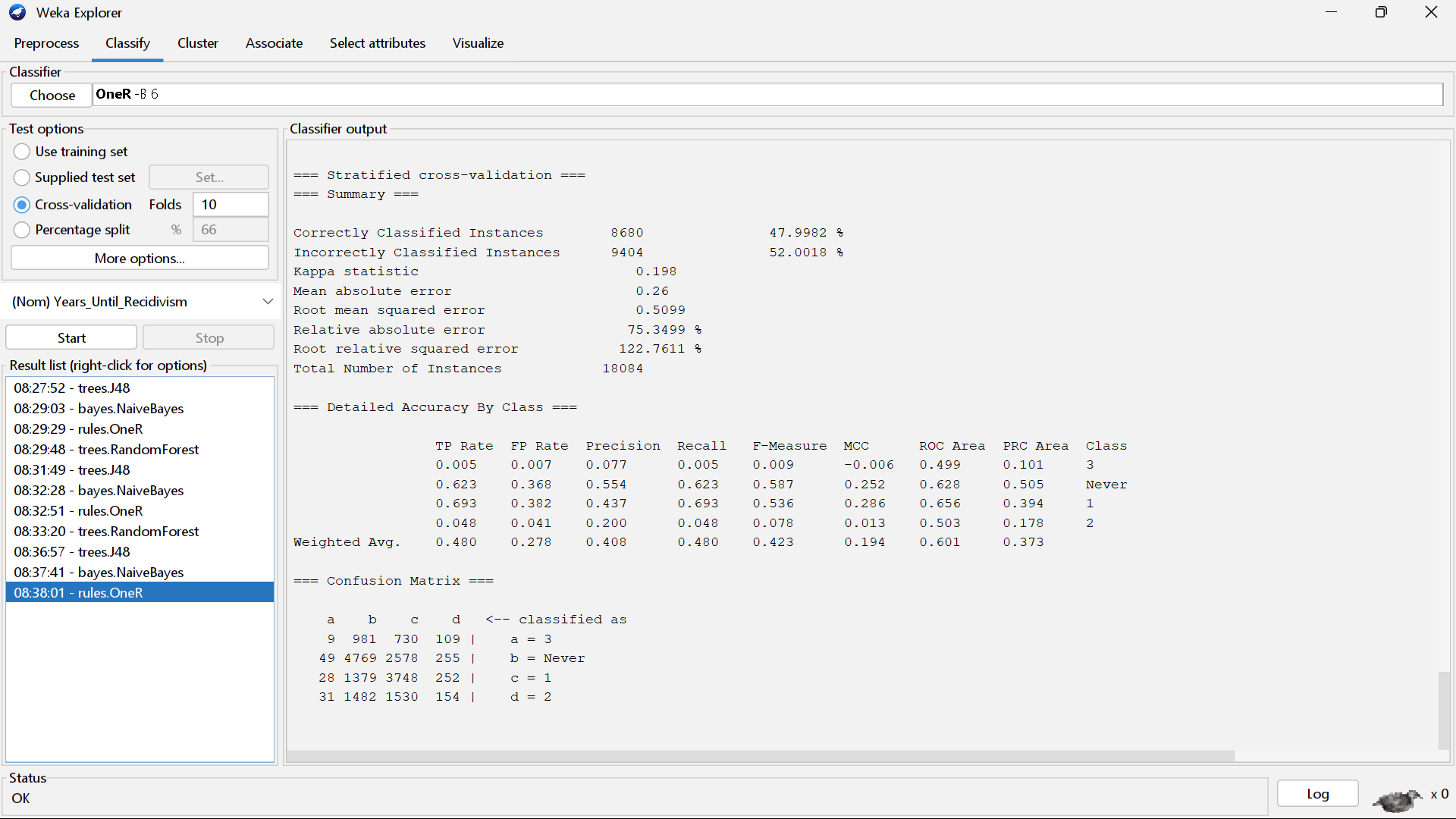
*Figure 14: J48 OneRAttributeEval Correlation Result*

Naive Bayes – OneRAttributeEval Attributes



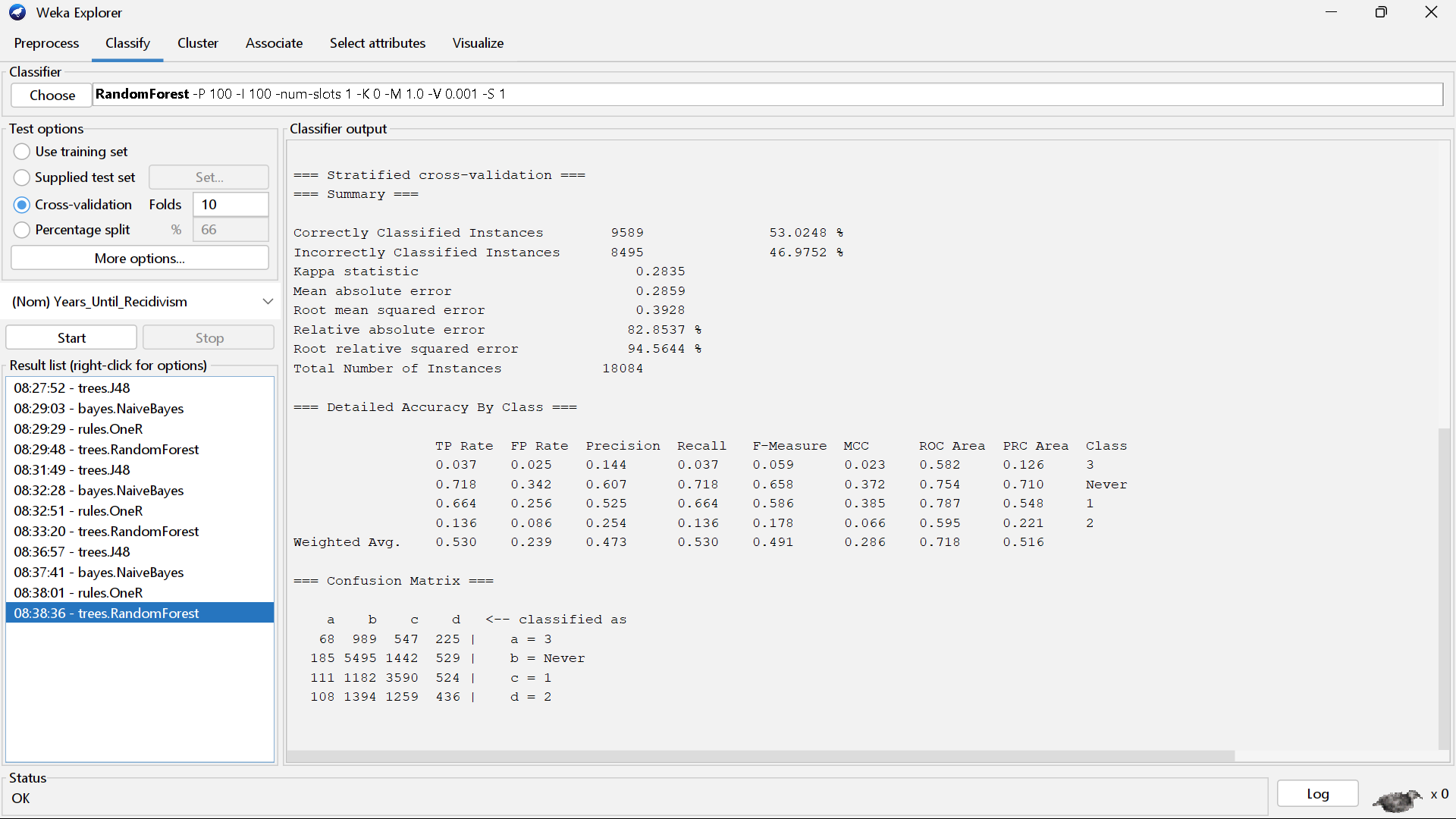
*Figure 15: NaiveBayes OneRAttributeEval Correlation Result*

OneR – OneRAttributeEval Attributes



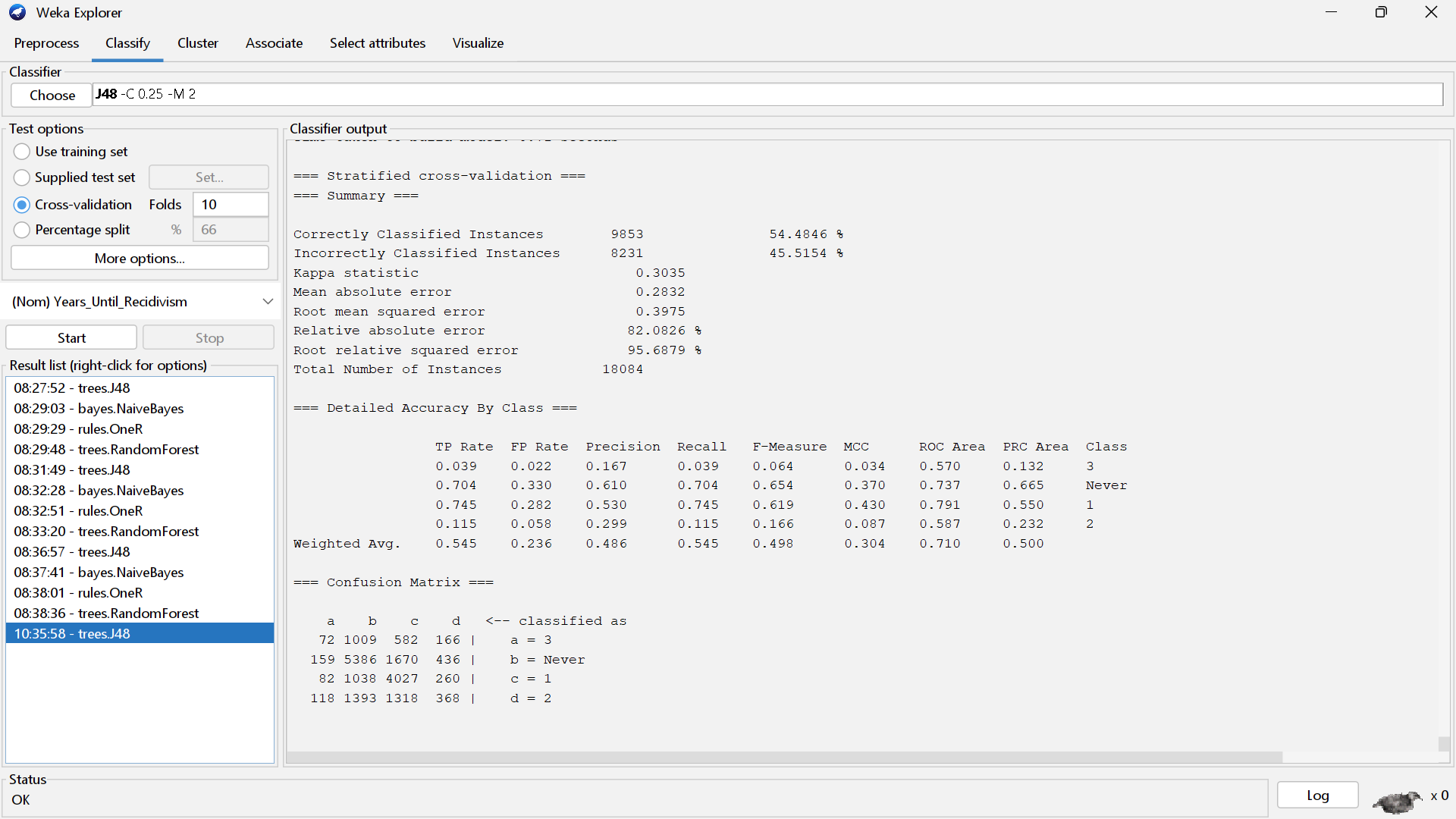
*Figure 16: OneR OneRAttributeEval Correlation Result*

RandomForest – OneRAttributeEval Attributes



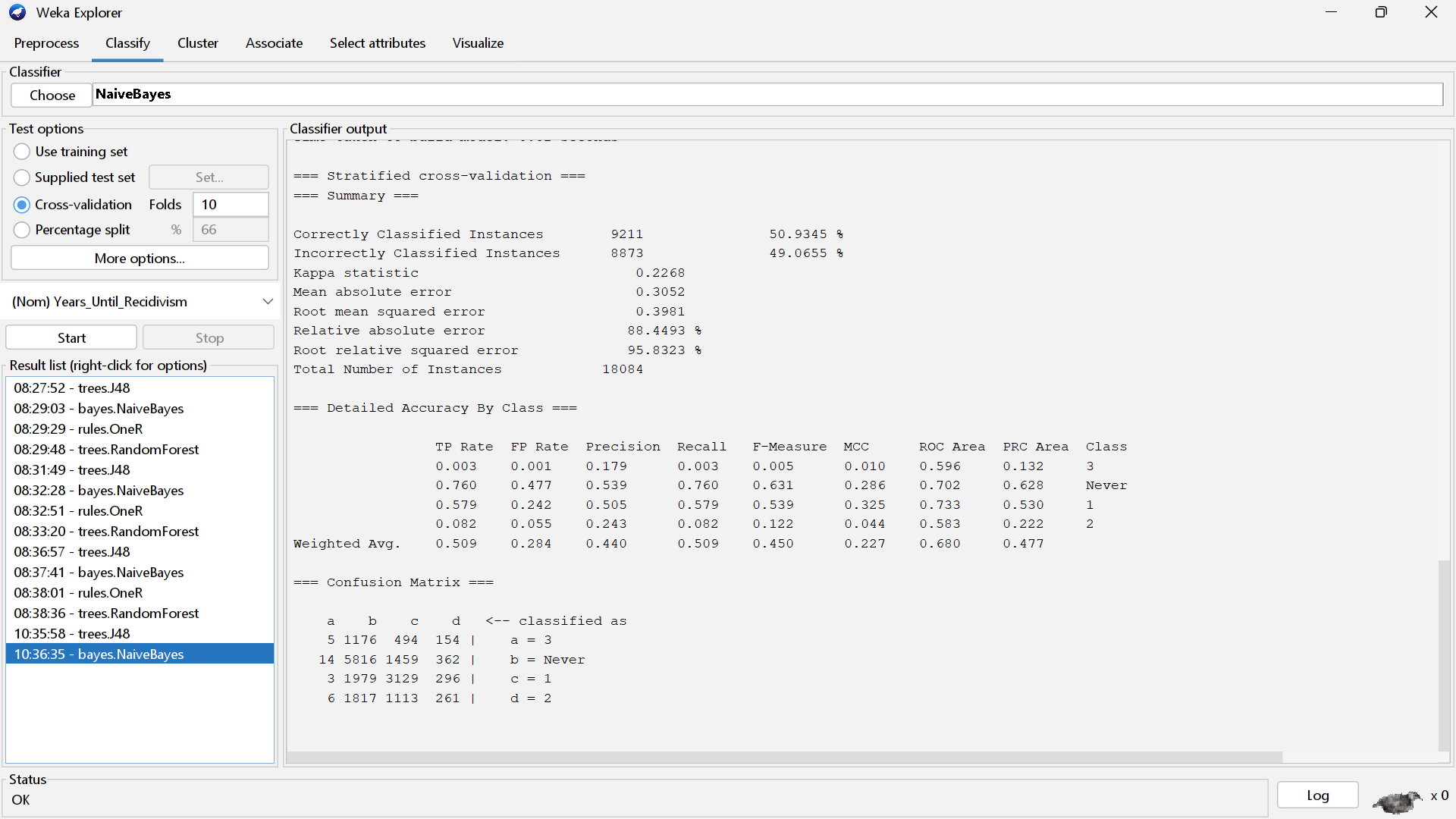
*Figure 17: RandomForest OneRAttributeEval Correlation Result*

J48 – WrapperSubsetEval Attributes



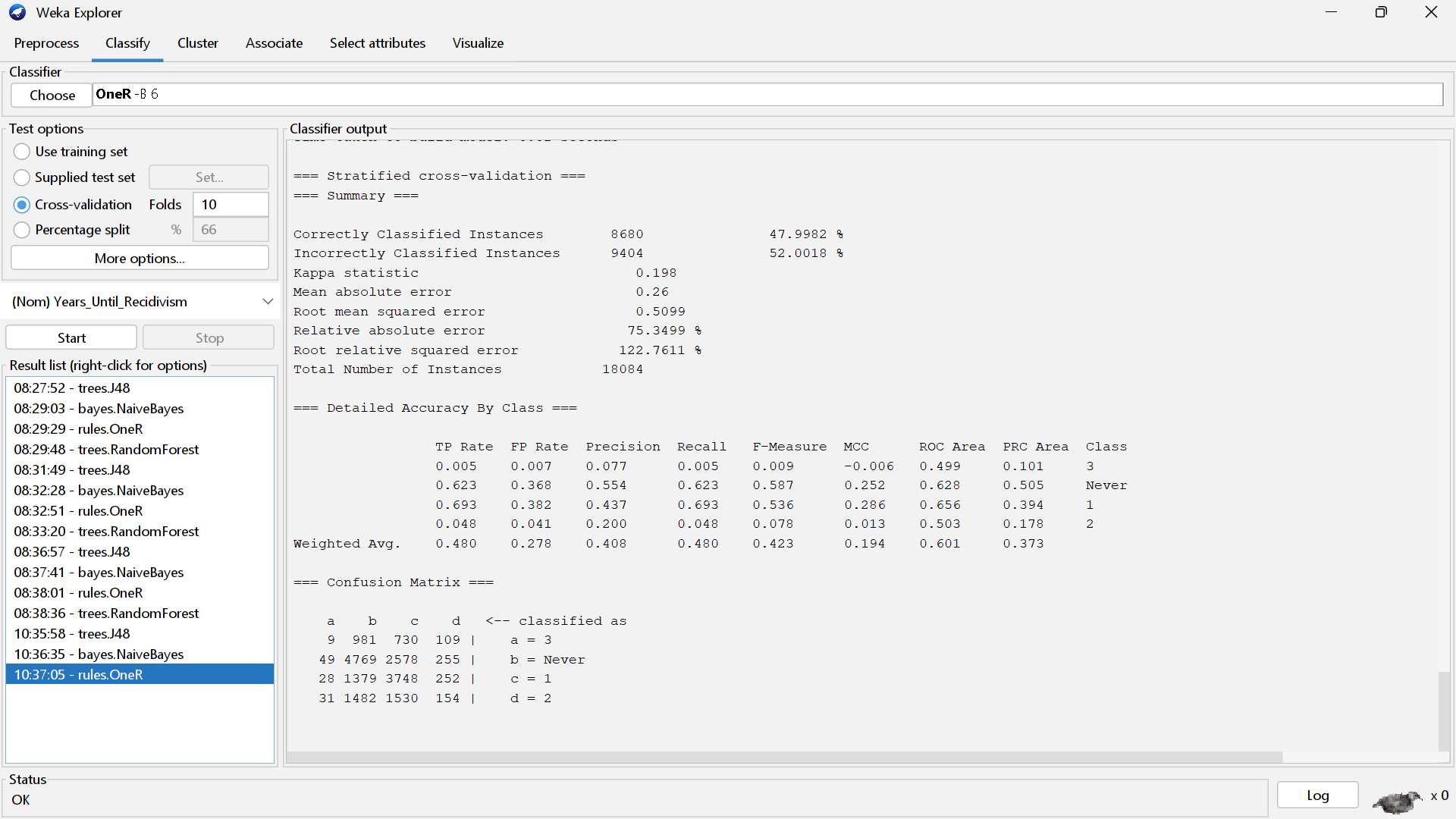
*Figure 18: J48 WrapperSubsetEval Correlation Result*

Naive Bayes – WrapperSubsetEval Attributes



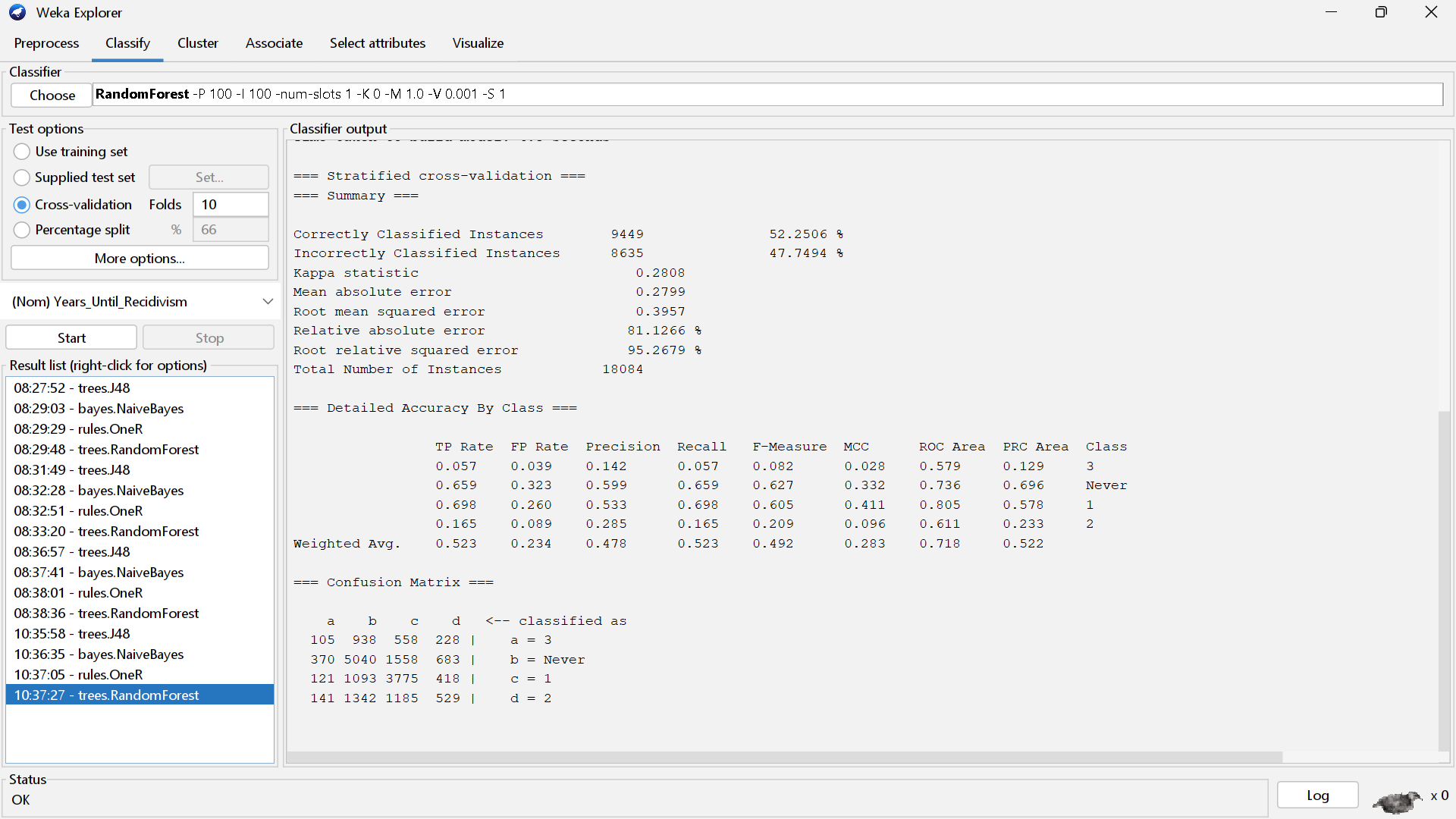
*Figure 19: NaiveBayes WrapperSubsetEval Correlation Result*

OneR – WrapperSubsetEval Attributes



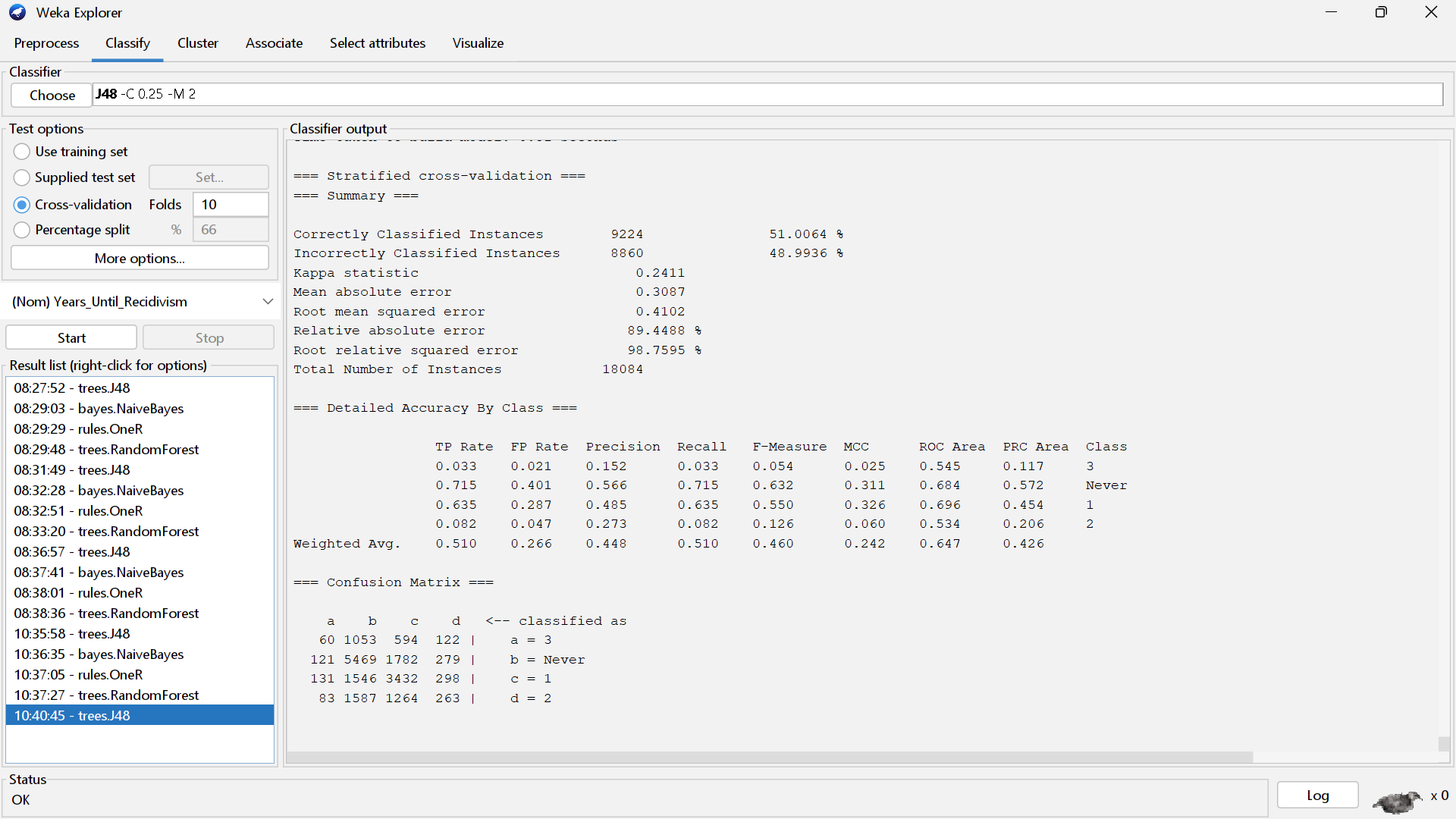
*Figure 20: OneR WrapperSubsetEval Correlation Result*

RandomForest – WrapperSubsetEval Attributes



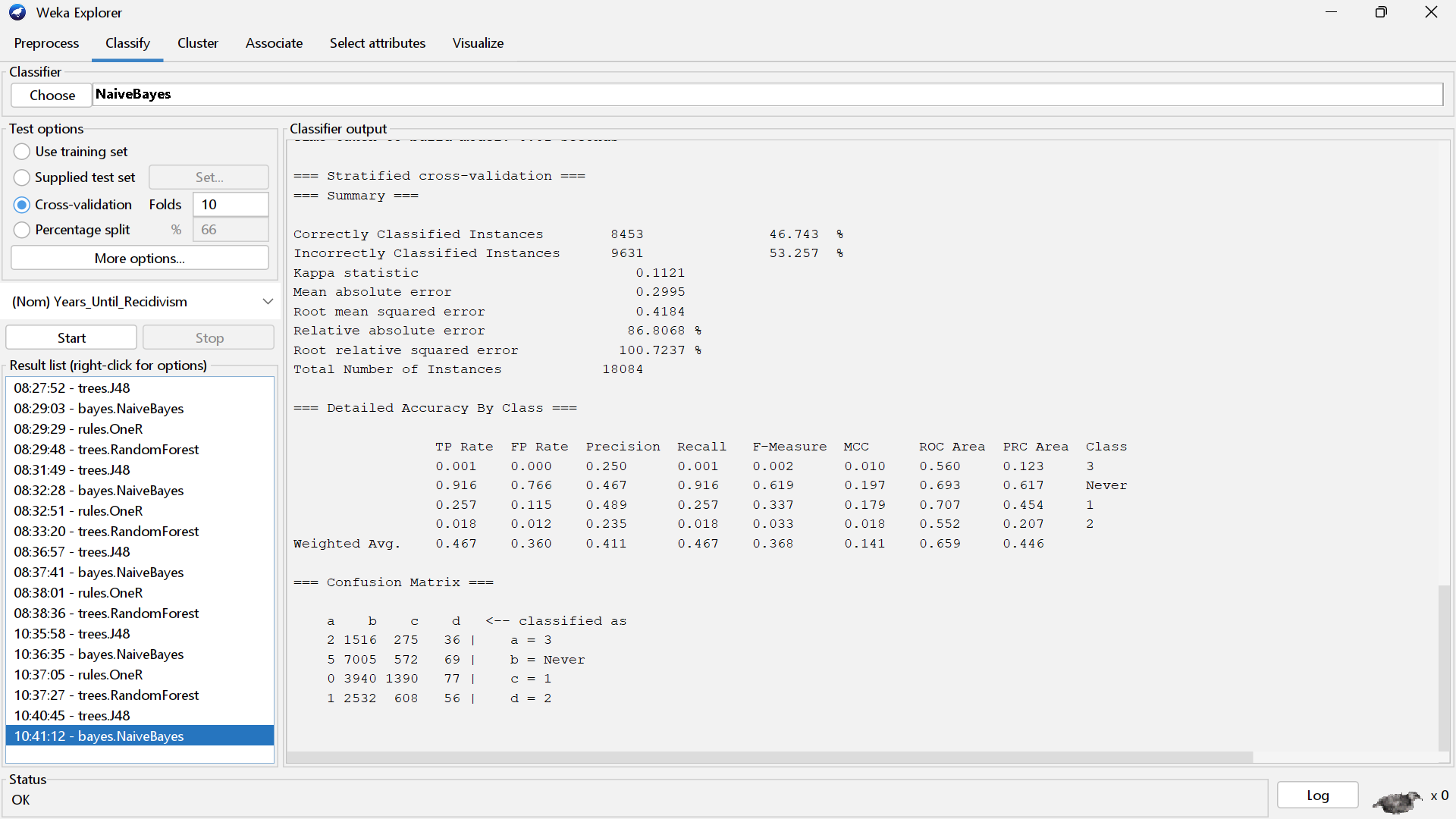
*Figure 21: RandomForest WrapperSubsetEval Correlation Result*

J48 – Self Selected Attributes



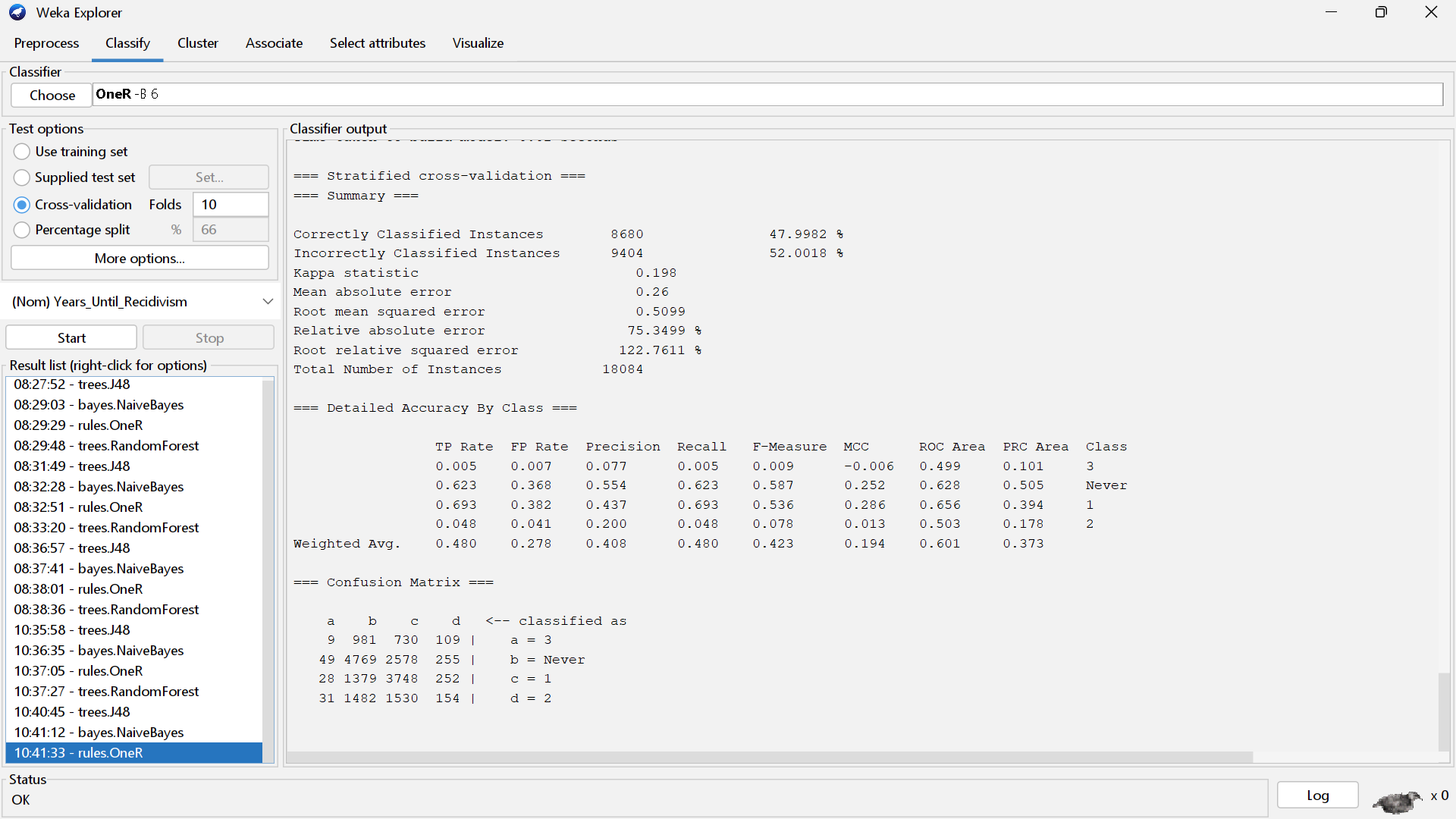
*Figure 22: J48 Self Selection Correlation Result*

Naive Bayes – Self Selected Attributes



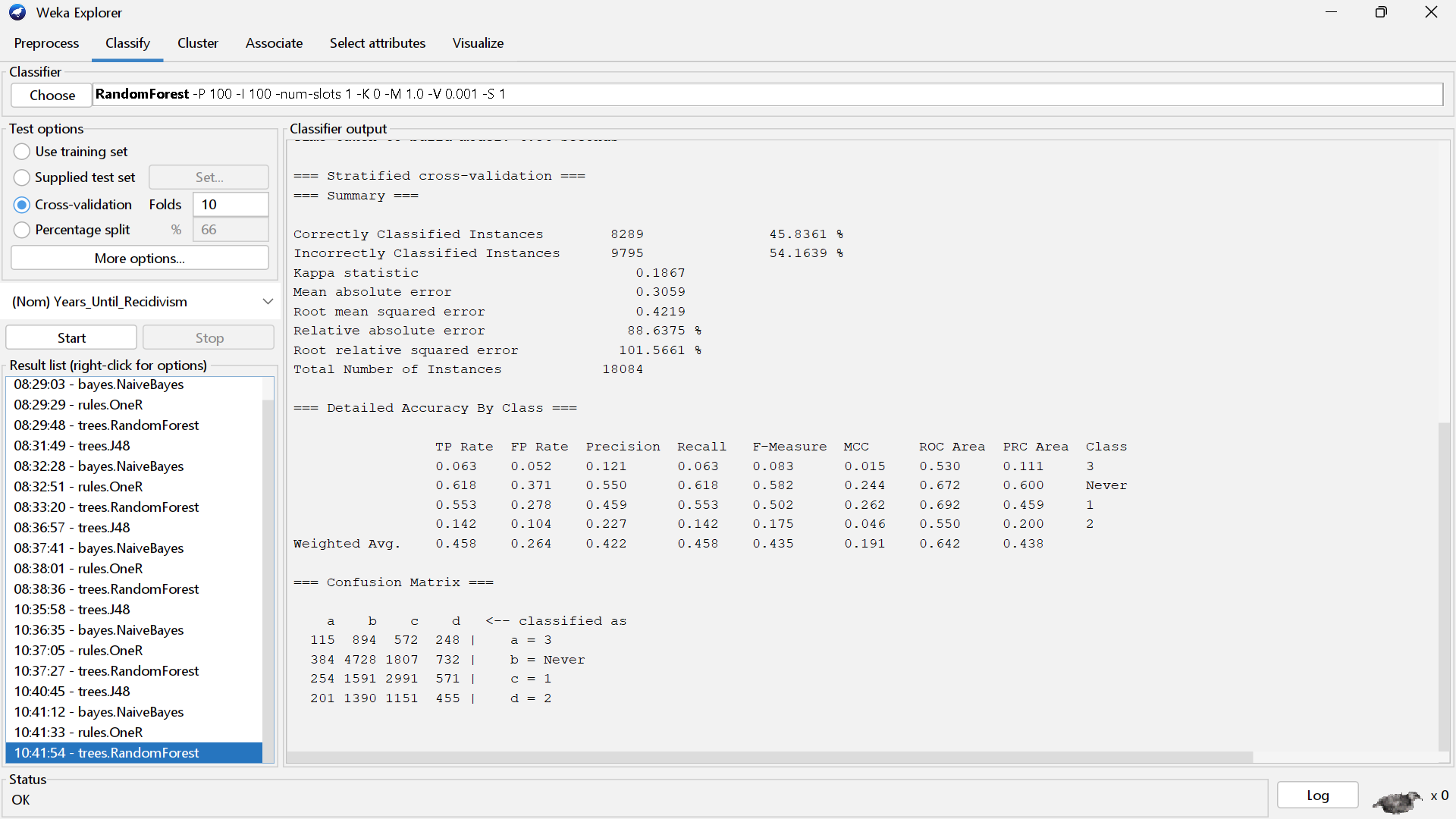
*Figure 23: NaiveBayes Self Selection Correlation Result*

OneR – Self Selected Attributes



*Figure 24: OneR Self Selection Correlation Result*

Random Forest – Self Selected Attributes



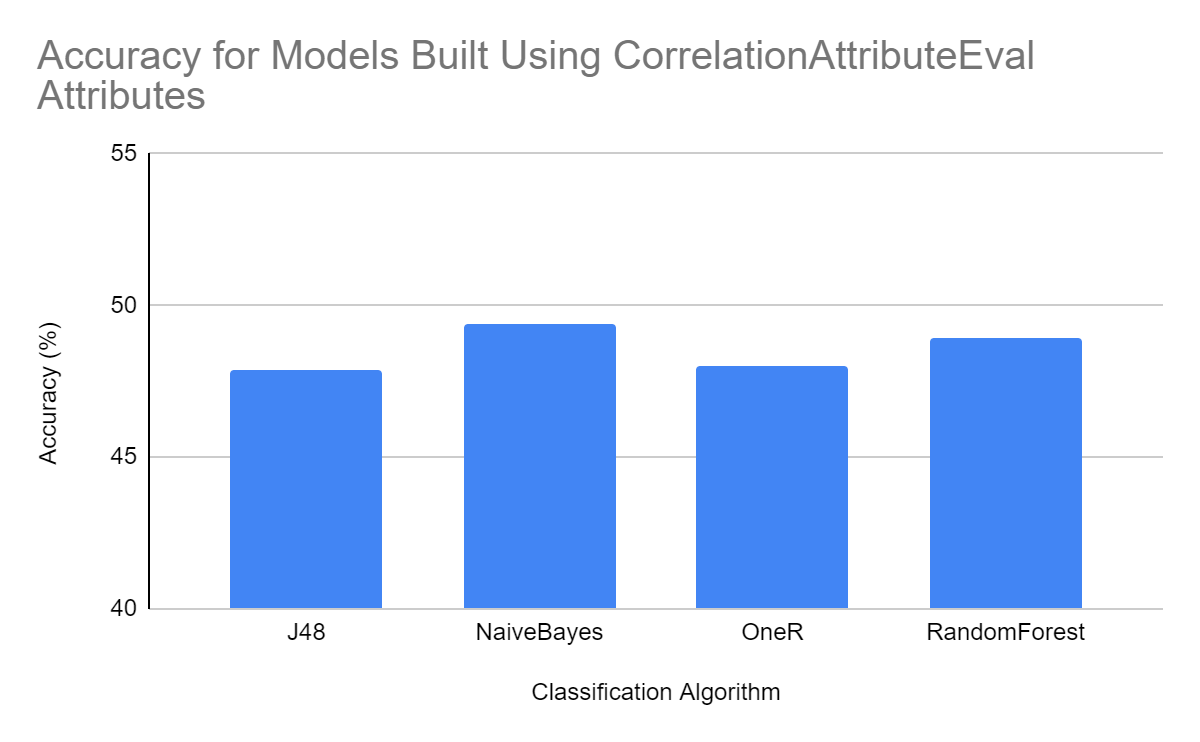
*Figure 25: RandomForest Self Selection Correlation Result*

# **Part 5 – Analysis & Conclusion**

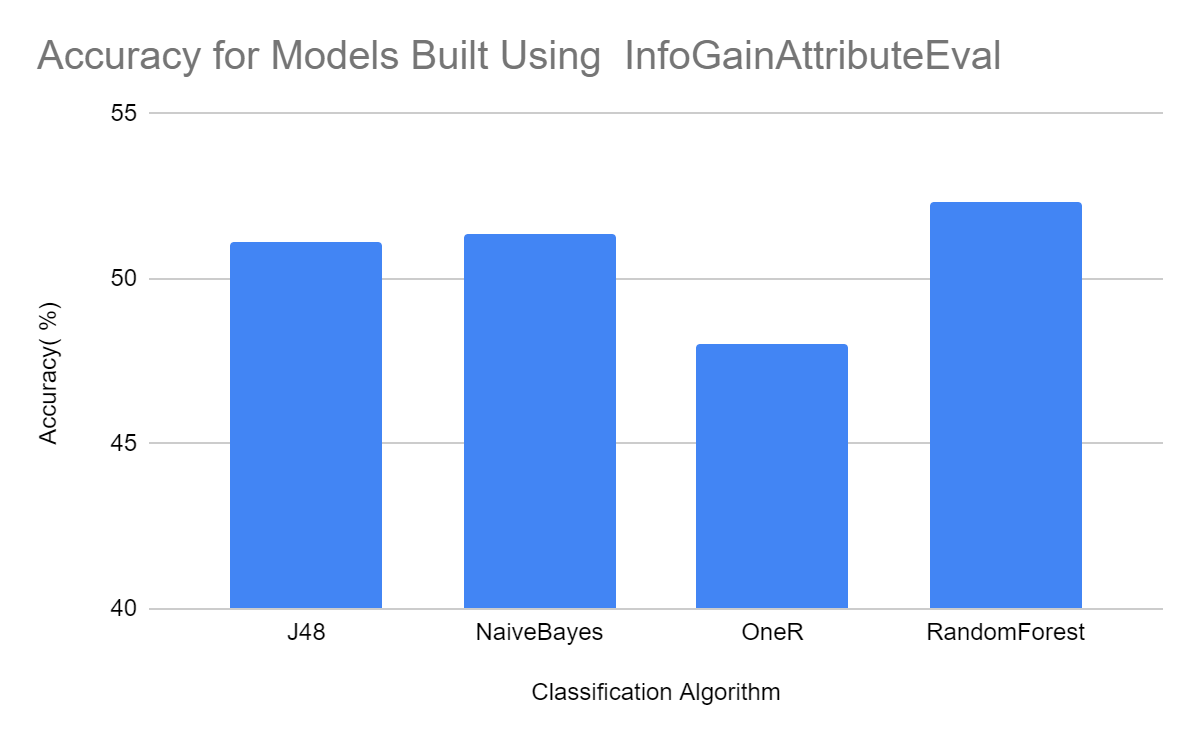
**Part 5.1 - Analysis**

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.

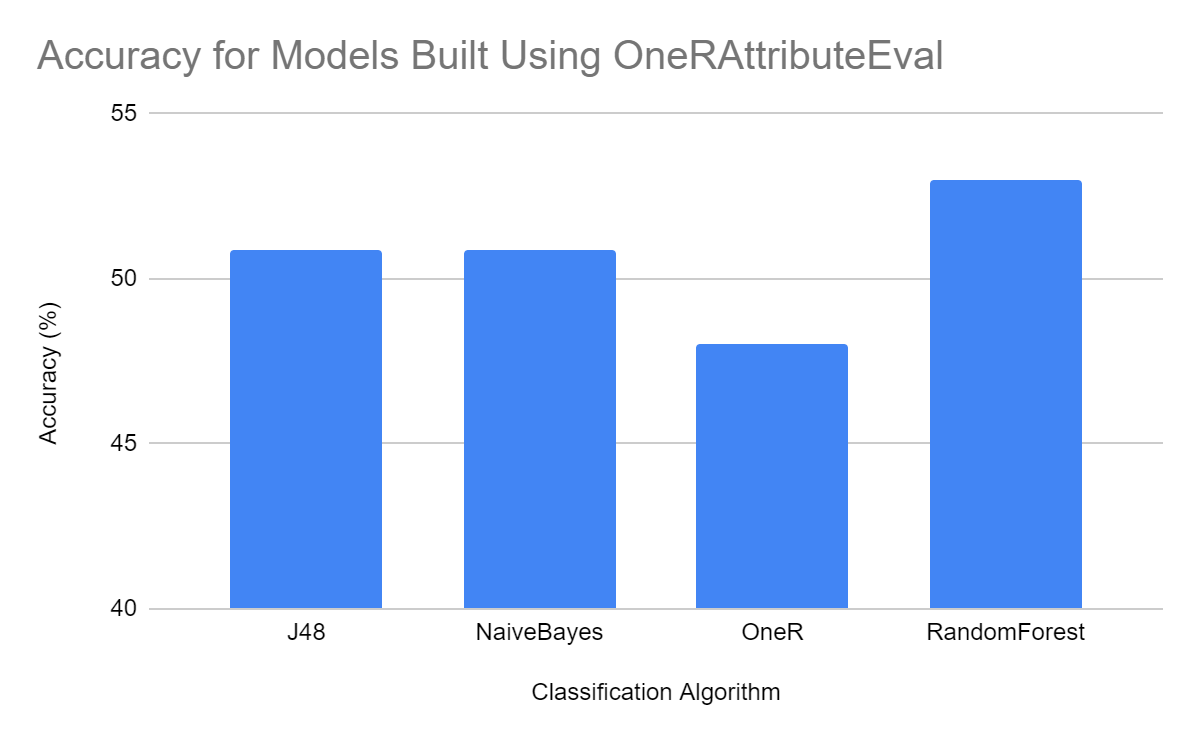
Below are five histograms, one for each set of attributes, that compare the accuracy of the four classification algorithms:



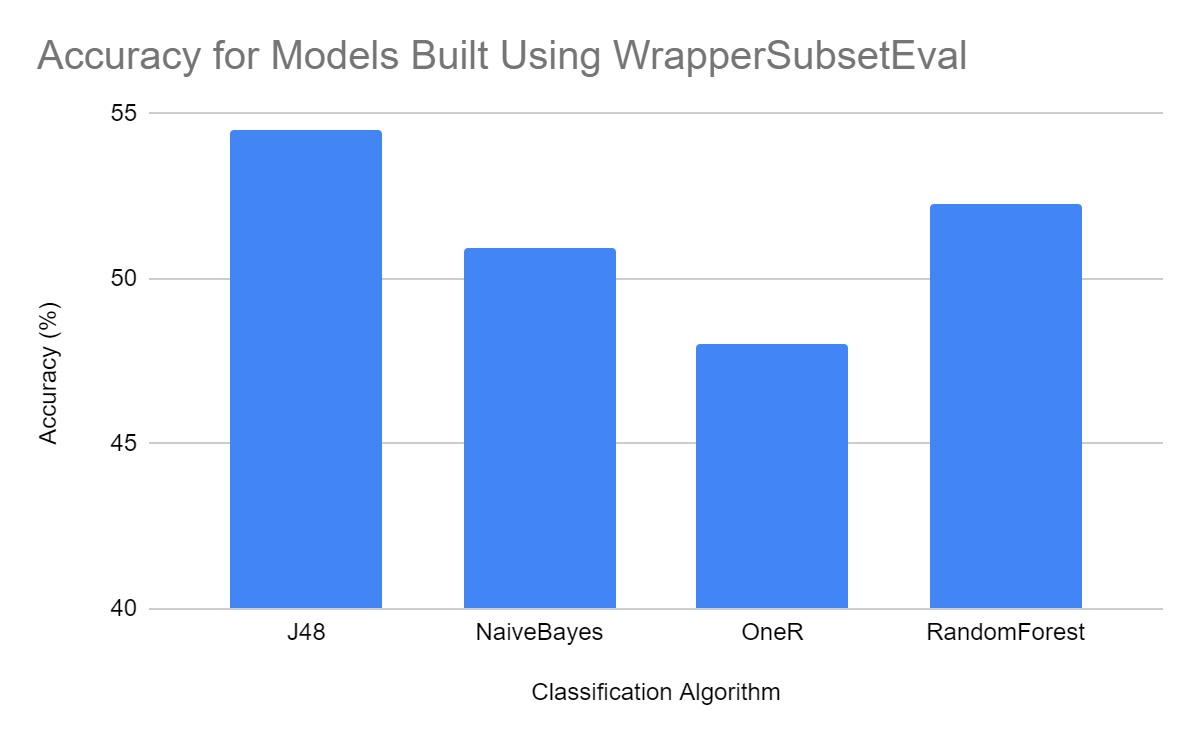
*Figure 26: Accuracy for Models Using CorrelationAttributeEval*



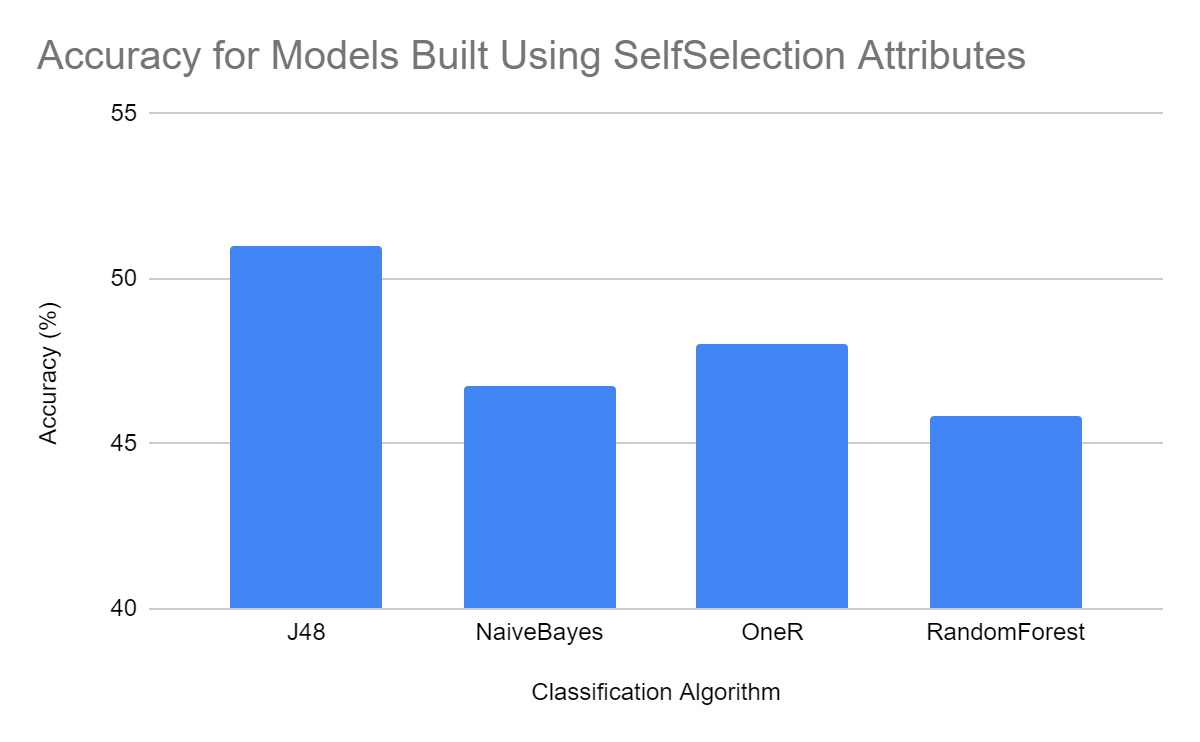
*Figure 27: Accuracy for Models Using InfoGainAttributeEval*



*Figure 28: Accuracy for Models Using OneRAttributeEval*



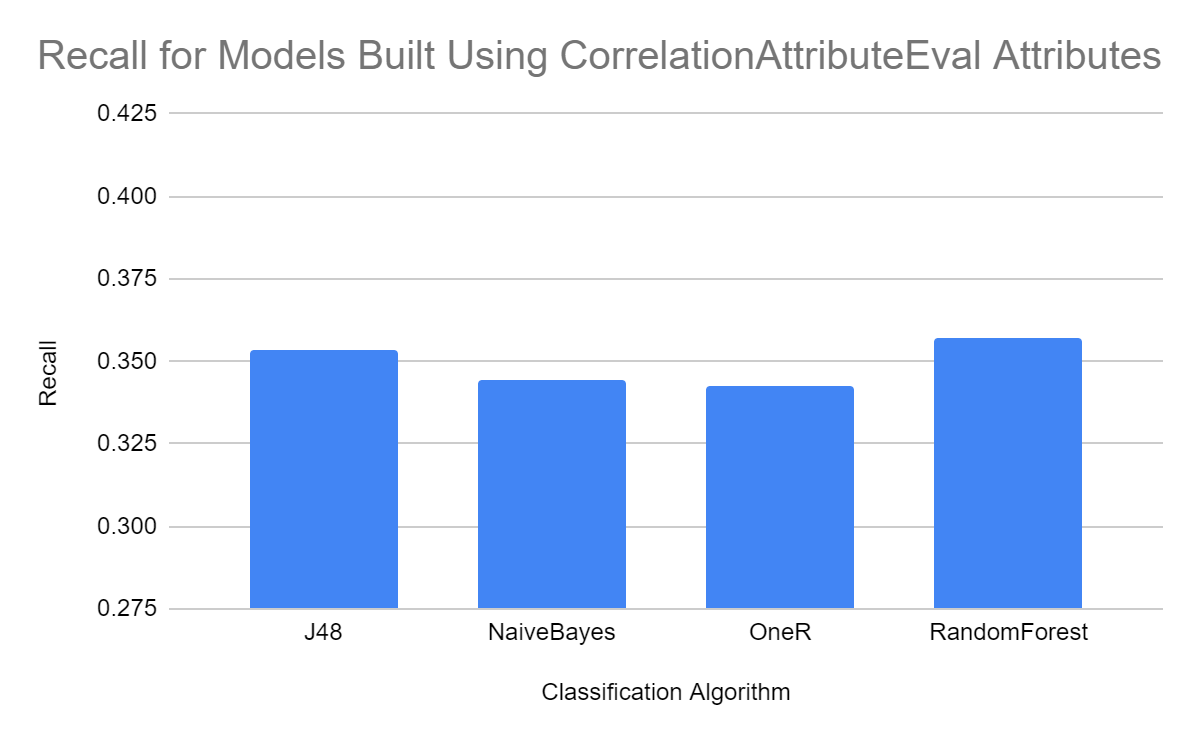
*Figure 29: Accuracy for Models Using WrapperSubsetEval*



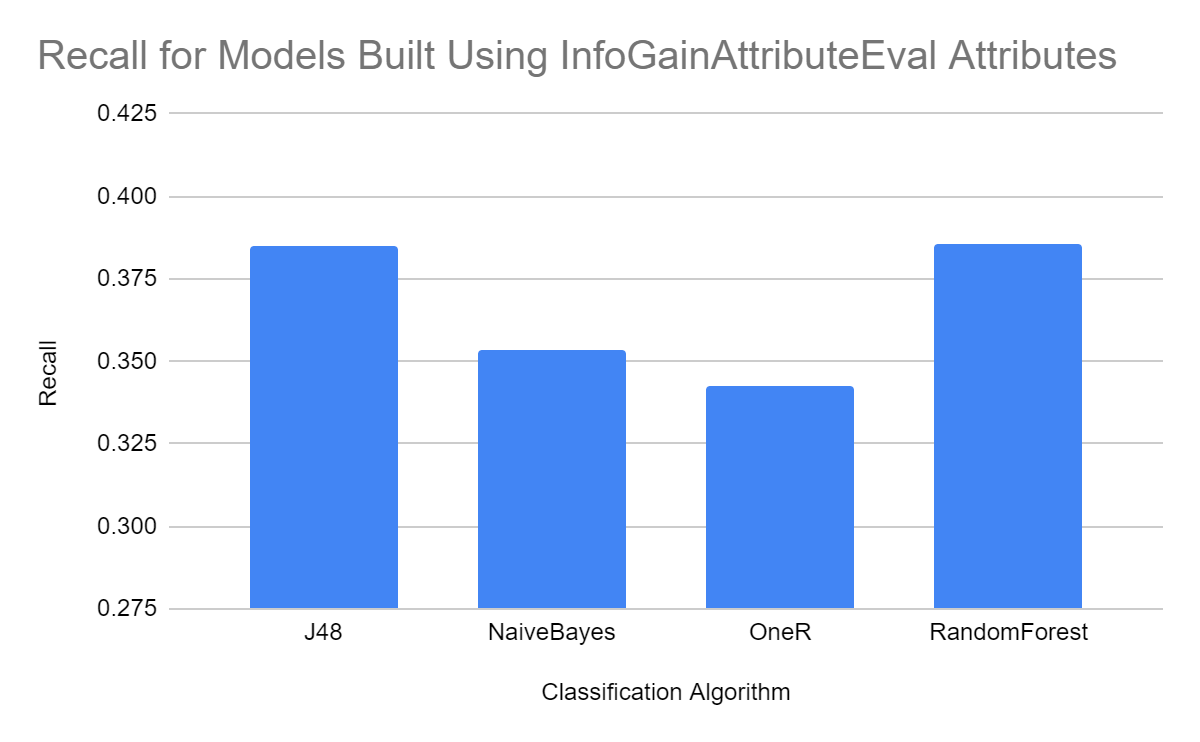
*Figure 30: Accuracy for Models Using Self-Selection*

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.

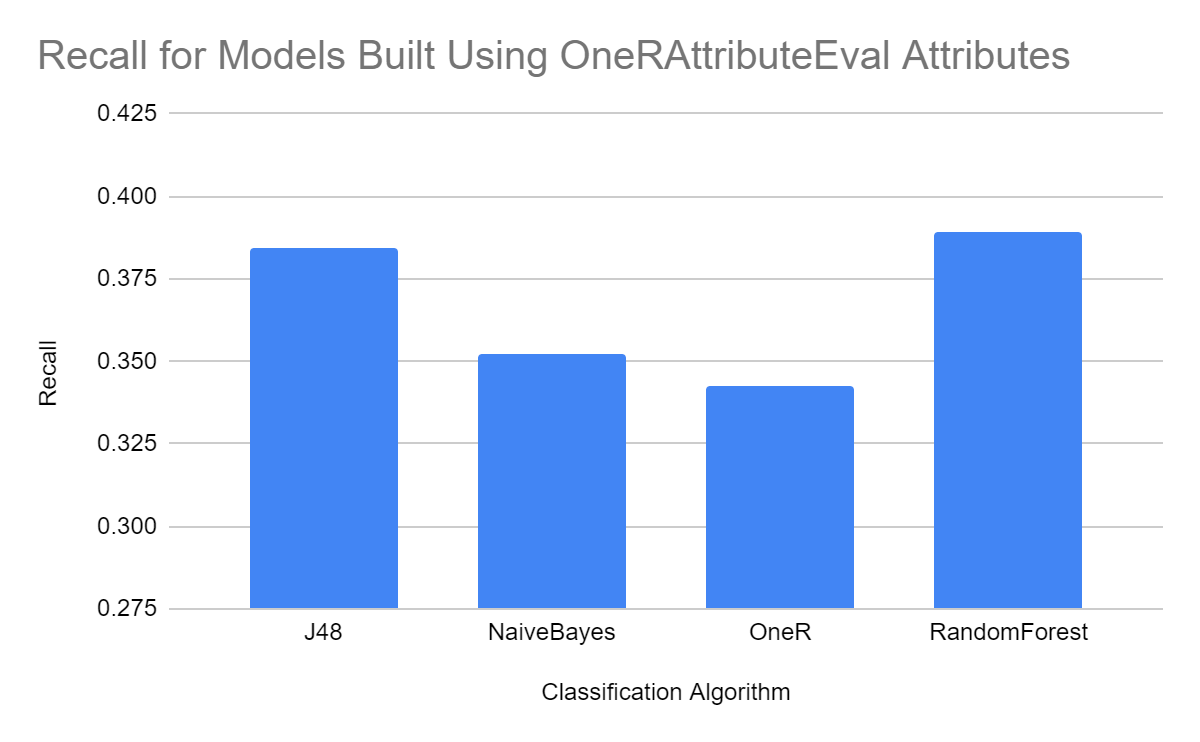
Similarly, 5 histograms are depicted in the following *Figures*. These histograms depict the recall for each model. We calculated recall using Macroaveraging due to the class imbalance in our dataset.



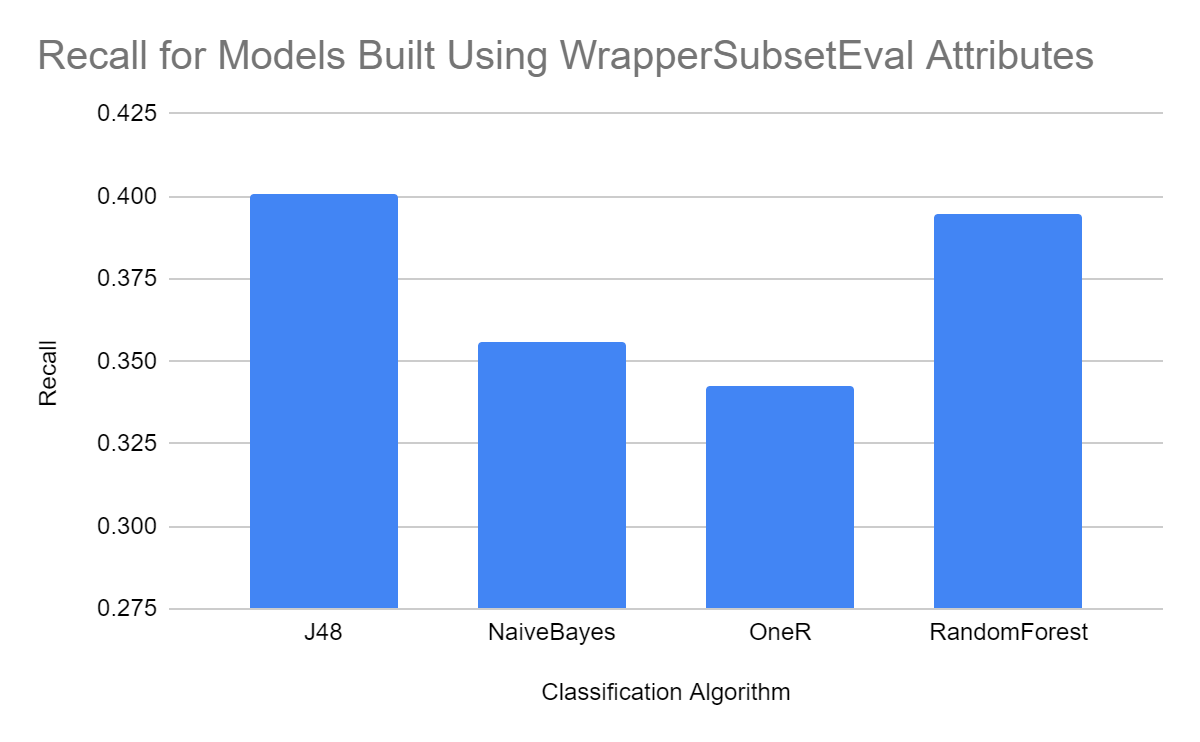
*Figure 31: Recall for Models Using CorrelationAttributeEval*



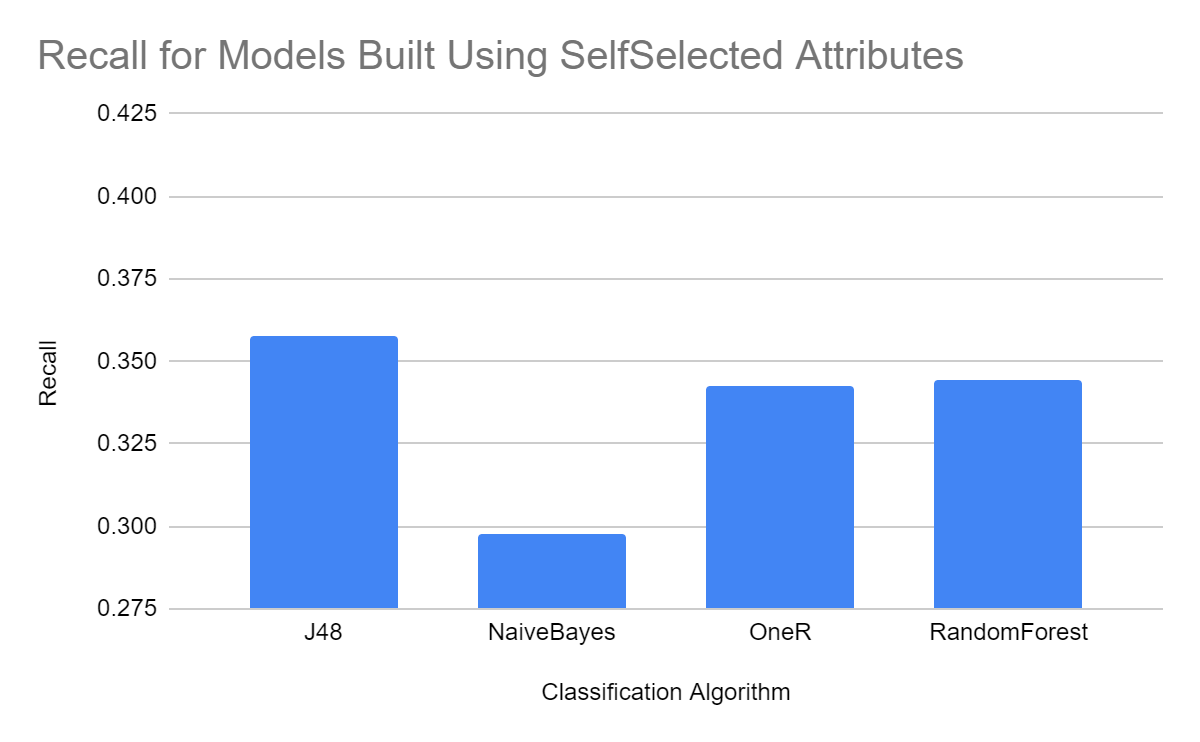
*Figure 32: Recall for Models Using InfoGainAttributeEval*



*Figure 33: Recall for Models Using OneRAttributeEval*



*Figure 34: Recall for Models Using WrapperSubsetEval*



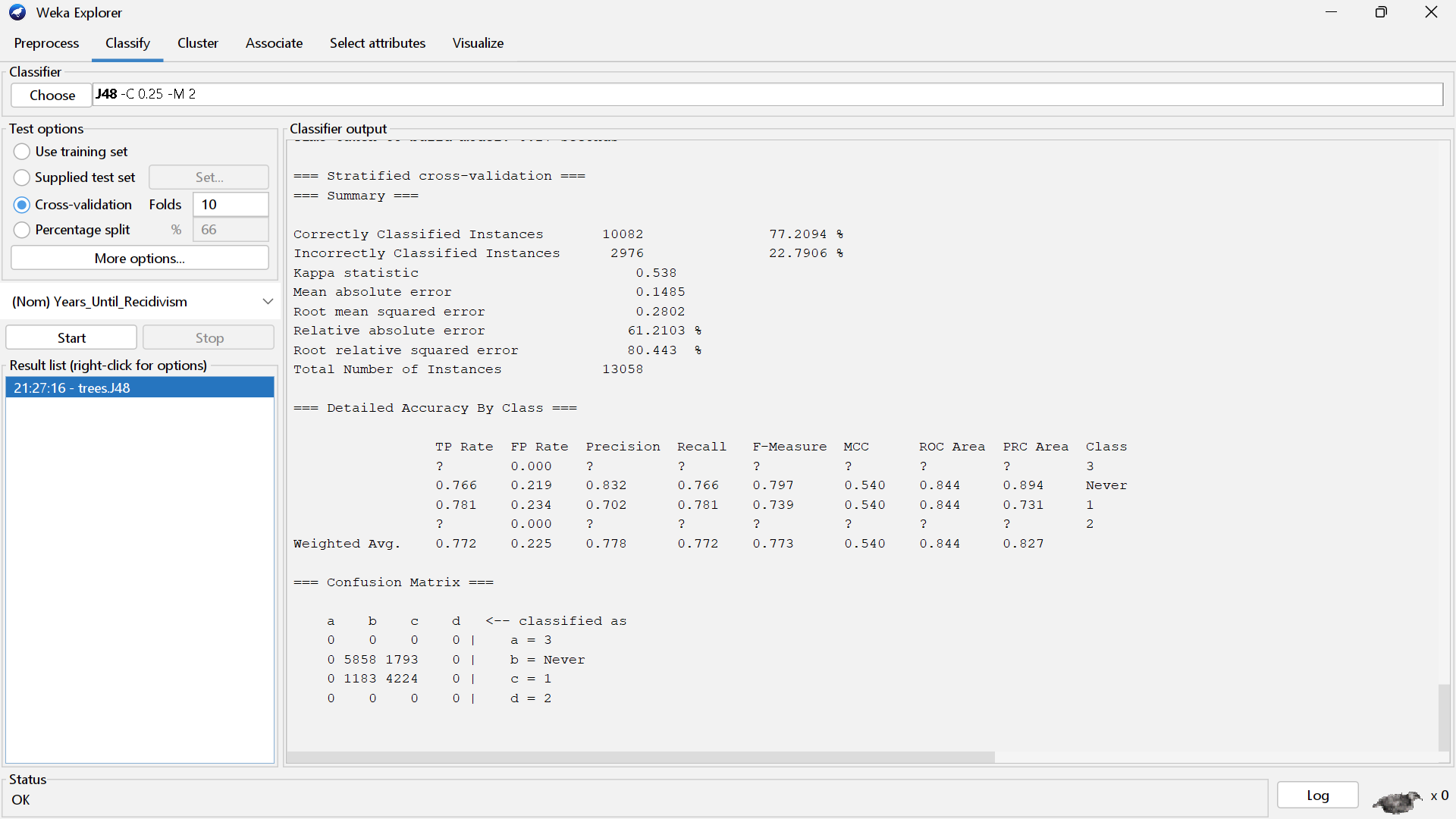
*Figure 35: Recall for Models Using Self Selected Attributes*

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.

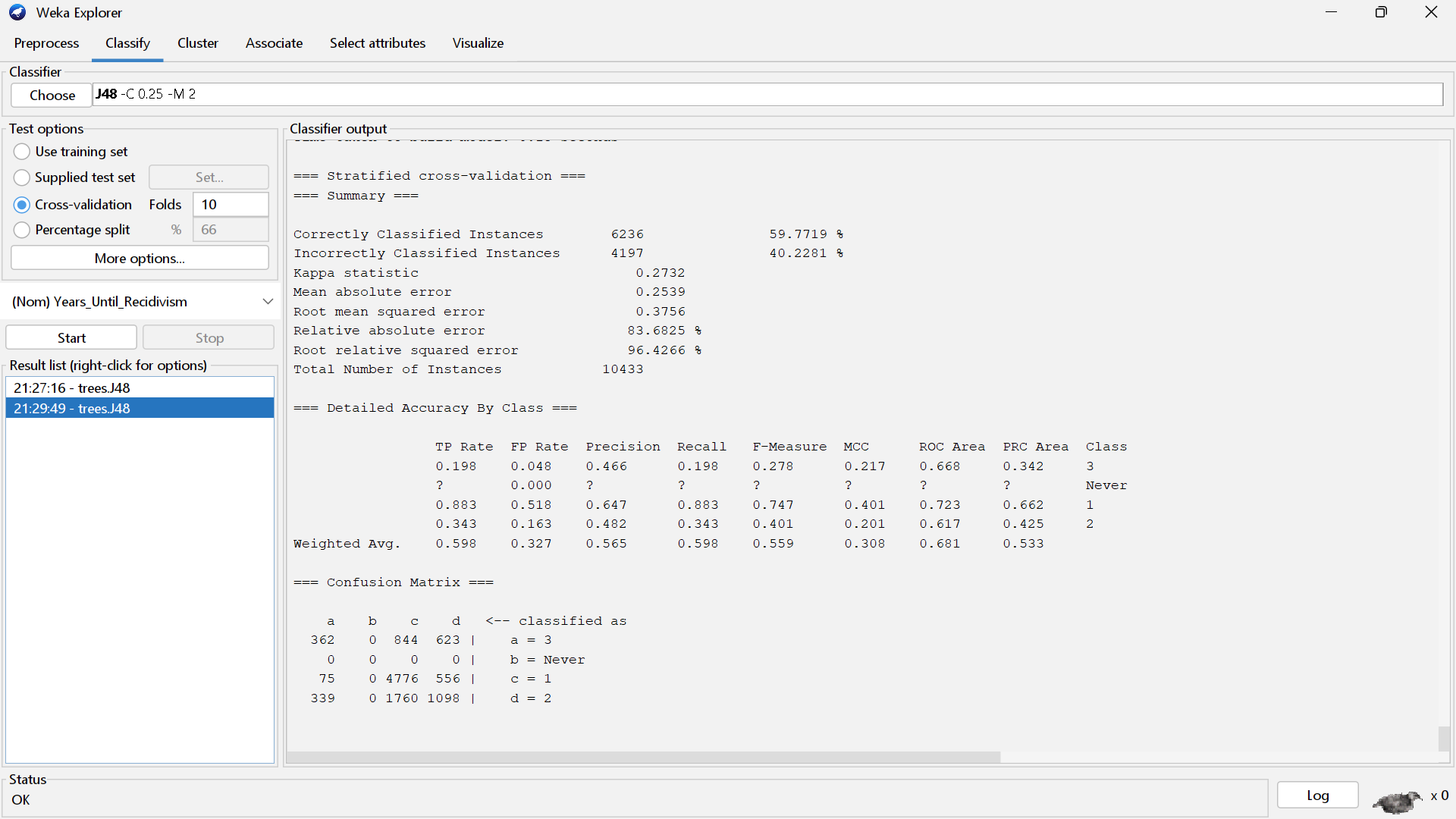
**Part 5.2 – Discussion of Error and Conclusion**

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.