Predicting Electoral Results Using Census and RDH Data

Deven Hagen and Vibby Janardhan

Dr. Yilmaz - Machine Learning 1

21 October 2024

**Statement/Project Goal**

Elections are the backbone of representative democracy—they determine the officials who make the important decisions for our country. Electoral analysis has many applications, from understanding demographic behavior to identifying thematic shifts in public opinion. Accordingly, our project aims to utilize demographics, housing data, past electoral results, and other related attributes to improve our understanding of how different groups vote; using Virginia precinct-level data, we will predict the party alignment of voting districts in the 2020 Presidential Election. This data will be useful for politicians to better understand the desires of different constituencies they represent, and for citizens to obtain information about the state of various races.

**Description of Dataset**

In order to facilitate proper analysis, one must use a comprehensive dataset. We acquired our dataset from the Redistricting Data Hub (RDH), which has various datasets based on the Census and other official government projects (see source 1).

In particular, we used the “va\_pl2020\_official\_vtds.csv” file from the download above. This file includes various critical variables, such as racial makeup, land area, housing units, and previous electoral results. The final dataset, before preprocessing, includes 463 attributes in total. We created a class attribute titled “WINPARTY” from the “PRES2020D” and “PRES2020R” columns, which represent the party that received the most votes in the 2020 election—each precinct will have a value of either “D”, representing the Democrats, “R”, representing the Republicans, or “T”, representing a tie. The dataset has 2465 instances, corresponding to each precinct in Virginia. In preprocessing, we will cut down the number of attributes significantly using attribute selection techniques such as Correlation and others to avoid the curse of dimensionality.

The dimension of our dataset currently is 462, because there are 463 attributes but the class does not count. The class distribution is as follows: 1747 “R”, 697 “D”, and 21 “T”. A number of columns are entirely missing values, and some precincts have disguised missing values for key attributes as well, although Weka does not show these. The dataset is skewed towards Republicans, although most of the “D” precincts are heavily Democratic-voting and thus outliers. In terms of the definitions of the attributes, a complete catalogue can be found if you click on the “[Census and AutoBound Edge field definitions (.xlsx file)](https://www.virginiaredistricting.org/2021/Data/ref/field%20definitions.xlsx)” link (see source 2). Although it is quite unfeasible to provide a detailed explanation of all 463 attributes, here is a representative sample:  
GEOID20: The unique Census geo ID that corresponds to each precinct.

ALAND20: The land area of the precinct.  
TAPERSONS: The total population of the precinct.

TAWHITEALN: The total White population in the precinct.

TABLACKALN: The total Black population in the precinct.

TAAMINDALN: The total American Indian and Alaskan Native population in the precinct.

TAASIANALN: The total Asian population in the precinct.

TANHPOALN: The total Hawaiian and Pacific Islander population in the precinct.

TAOTHERALN: The total population of all other races not previously mentioned in the precinct.

TAHOUSING: The total number of homes in the precinct.

PRES16D: The number of votes in the precinct for 2016 Democratic nominee Hillary Clinton.

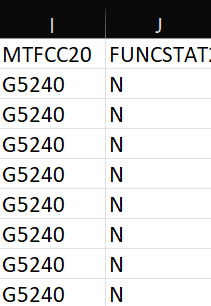
PRES16R: The number of votes in the precinct for 2016 Republican nominee Donald Trump.

There are more combinations and variations of attributes like these, but this provides a general idea of the attributes contained in our dataset.

**Intermediate Report - 10/7/24**

**Data Preprocessing**

The next steps were to pare down our 463 attributes into a manageable number and remove instances with many missing values. As stated in our proposal, many of the columns in our dataset had either missing values or identical values for each instance. It would take up too much space to enumerate each attribute removed, but below are examples:

An example of columns missing every value An example of columns with identical values

After getting rid of these, we were down to 369 attributes—still a lot! Each election in our dataset was split into two columns: the number of Democratic votes and the number of Republican votes. The next step was normalization, and many of our methods came from ML Slides 4 (see source 3). In order to normalize the data, since some precincts have more people than others, we transformed each of these into the percentage of votes for each party. For example, we transformed USSEN18D, USSEN18R, and USSEN18L into USSEN18DPERC, which represents the percentage of total votes received by Democrats. We then repeated this process on the remaining election results.

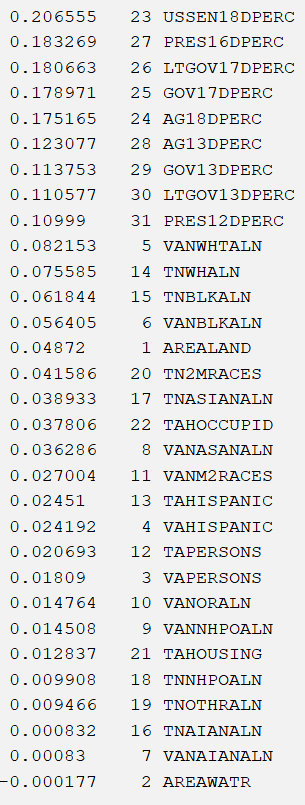
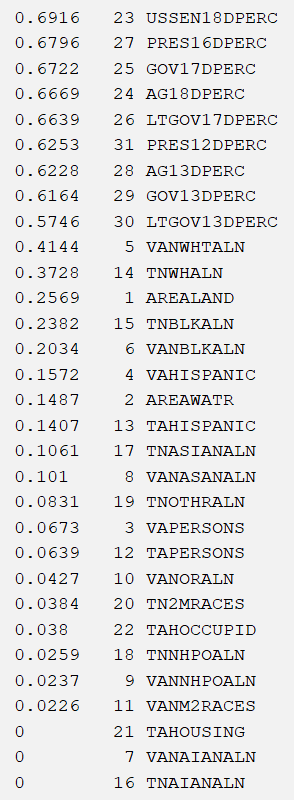
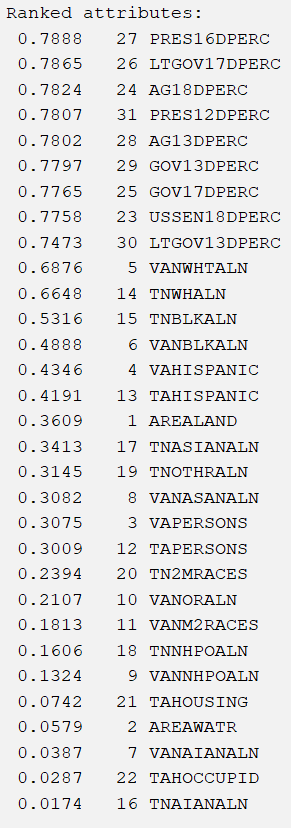
The vast majority of the attributes remaining were related to race - particularly obscure combinations of races. To address the excess columns for race data, we combined the multiracial columns into a new attribute representing multiracial people. This consolidation of columns helped avoid the curse of dimensionality significantly. We had both the Voting Age and Total numbers for racial demographics, and we kept both to ascertain whether key patterns would emerge. Similarly to what we did with electoral results, we normalized the race data as percentages of the total population.

After normalizing race and electoral attributes, some null values became apparent due to a lack of data. Since these instances were missing very key values, particularly for past election data, we removed them from the dataset. After this removal, there were 2423 instances. We also deleted all instances with a class label of “T” (tied). Voters do not vote for a tie but instead choose a preferred candidate, so trying to predict precincts that will end in a tie is not a useful exercise. This left us with 2411 instances. When it came to the columns regarding housing, most were missing many values, so we kept it to just TAHOUSING, TAHOCCUPIED, and TAHVACANT, removing the others. Since TAHOCCUPIED + TAHVACANT = TAHOUSING, we removed TAHVACANT, since it can be derived from the other attributes. We then normalized the housing data on a per-capita basis.

We used decimal scaling for AREALAND and AREAWATR to ensure their influence was not exaggerated, since some of the values for these attributes approached 10^11. Additionally, we used min-max normalization to scale VAPERSONS (voting-age population) and TAPERSONS (Total population). Finally, we changed our class labels from “R” and “D” to 0 and 1, where “R” is 0 and “D” is 1. Changing each value to a number will help when we use our models on the dataset. To conclude, after preprocessing, there were 32 attributes and 2411 instances left in our modified dataset.

**Attribute Selection**

To identify which attributes should be used in a classification model, four attribute selection methods were employed: OneRAttributeEval, CorrelationAttributeEval, InfoGainAttributeEval, and ReliefFAttributeEval. The results of each selection technique allowed us to accurately choose our attribute set. Each previous election result (USSEN18DPERC, AG18DPERC, GOV17DPERC, LTGOV17DPERC, PRES16DPERC, AG13DPERC, GOV13DPERC, LTGOV13DPERC, PRES12DPERC) consistently ranked high, so we will keep them. Beyond that, VANWHTALN (total only white voting-age population), VANBLKALN (total voting-age black population), and AREALAND (total land area) also consistently ranked high. Therefore, the above 13 attributes were chosen for the model. The results for each attribute selection method can be seen below.



**OneR Correlation InfoGain ReliefF**

**Train-Validation-Test vs K-fold Cross-Validation**

We used Pandas to read our new dataset into Python as a Dataframe. We decided to make our training set 70% of the instances, our validation set 15%, and our testing set 15%. We then used Scikit-Learn’s train-test split twice to create our sets, making sure to use the stratify parameter to keep the balanced class distributions. We took some of this code from the ML Slides 2 presentation (see source 4). These new datasets can be found in our drive: each attribute selection algorithm has a folder with the three datasets. Here is the code we used:

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import os

for folder in ("CORRELATION", "INFOGAIN", "ONER", "RELIEFF"):

os.chdir(folder)

df = pd.read\_csv(f"{folder}\_DATASET.csv")

df\_train, df\_test\_val = train\_test\_split(df, test\_size=0.3, stratify = df["WIN\_PARTY"], random\_state=42)

df\_test, df\_val = train\_test\_split(df\_test\_val, test\_size=0.5, stratify = df\_test\_val["WIN\_PARTY"], random\_state=42)

df\_train.to\_csv('train.csv', index=False)

df\_test.to\_csv('test.csv', index=False)

df\_val.to\_csv('val.csv', index=False)

os.chdir('..')

However, we also tested implementations of K-fold cross-validation through Weka and decided that it was superior to simply creating train-validation-test sets due to its more comprehensive validation methods and simple implementation.

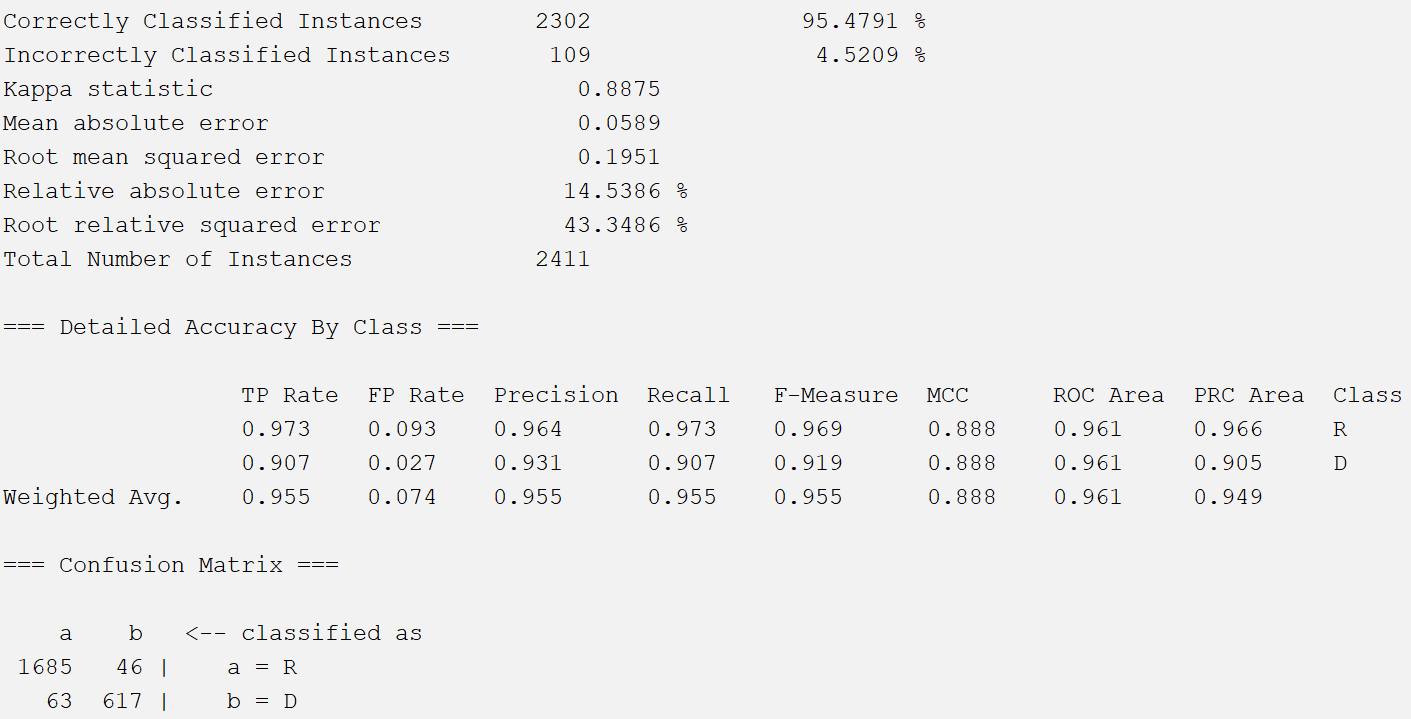
We soon realized when we opened WEKA that we had to switch our class labels from “0” and “1” to “R” and “D” because WEKA was reading the 0 and 1 as floats and hence the Naive Bayes classifier was not functioning.

**Results and Evaluation**

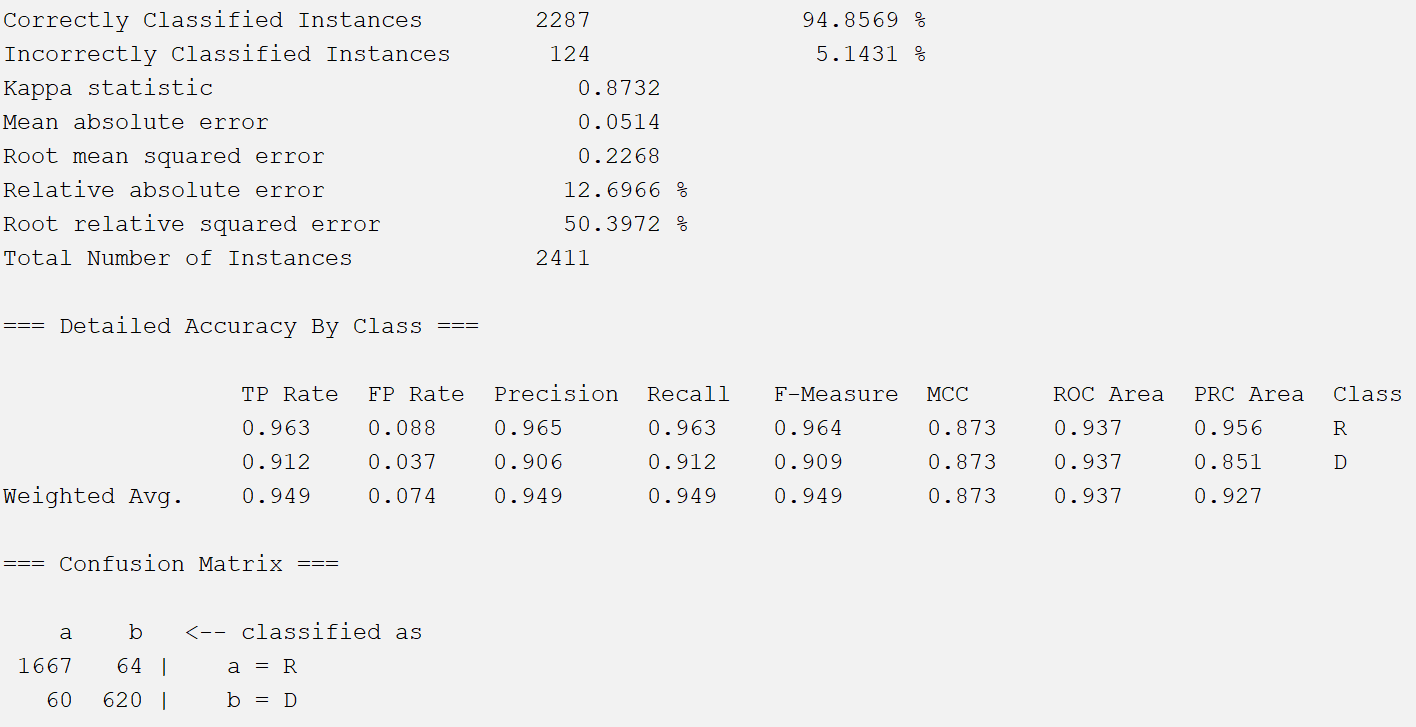
For each dataset—OneR, Correlation, InfoGain, ReliefF, and our own chosen one—we applied four models to classify the class variable: J48 decision tree, Naïve Bayes, OneRClassification, and K-nearest neighbor. We used a K-fold value of 10 for each model, as we found changing it from this value did not significantly improve the model. The screenshots of the results are below.

***OneR Dataset***

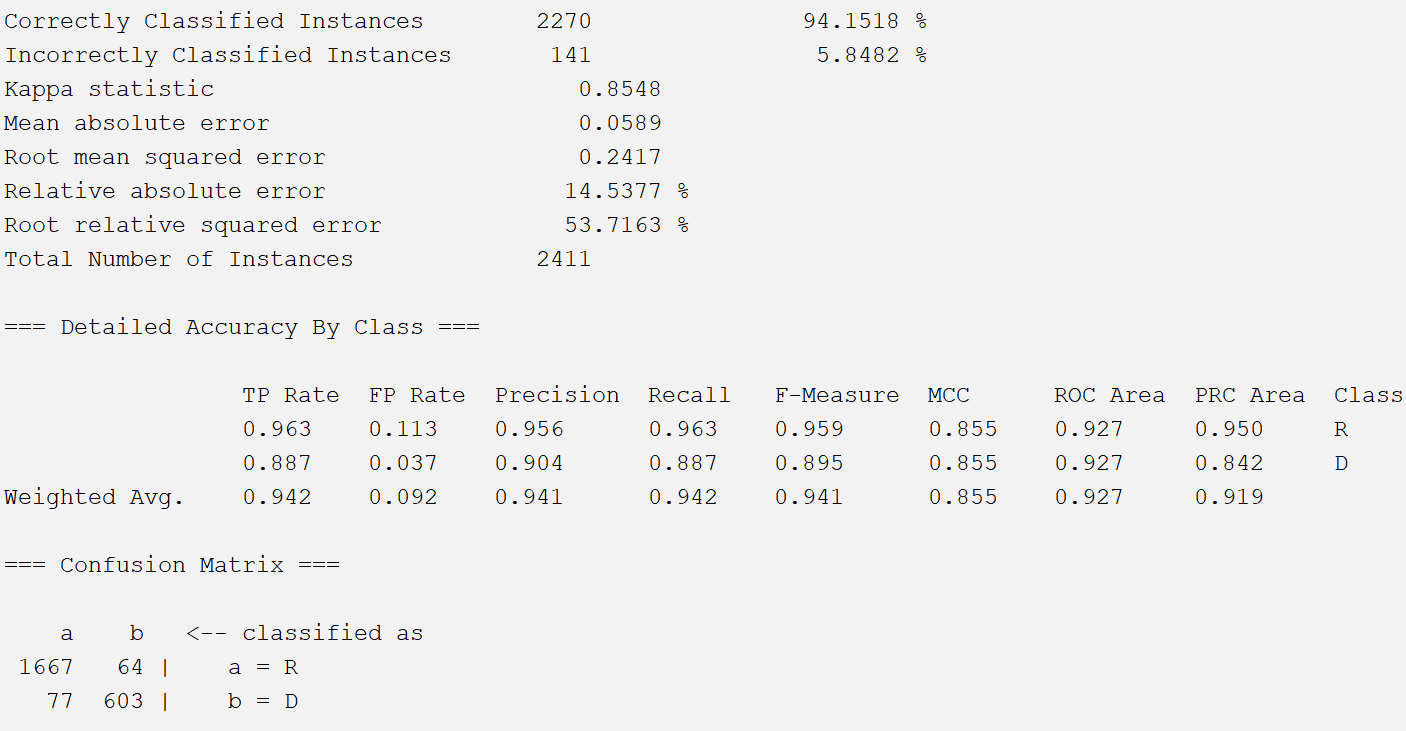
J48 decision tree: Accuracy = 0.954791, TP = 0.955, FP = 0.074, ROC = 0.961



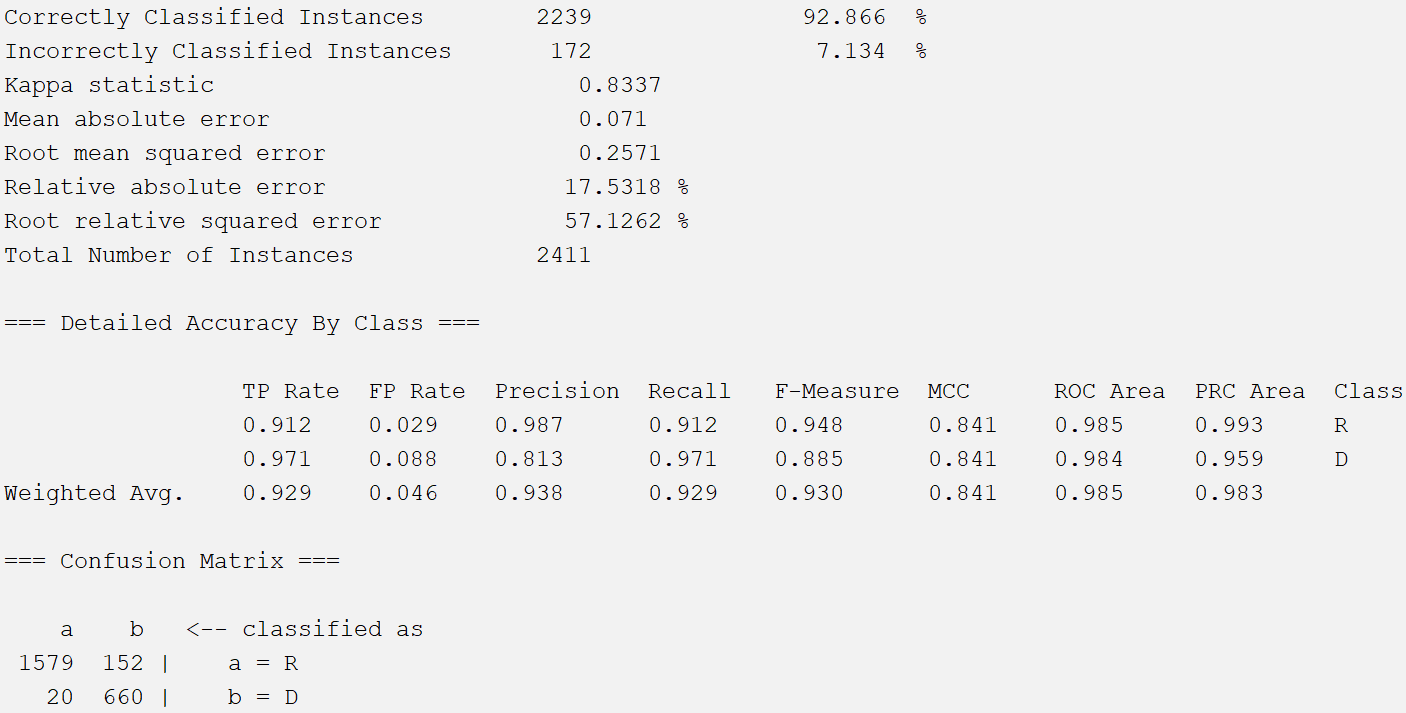
OneRClassifier: Accuracy = 0.948569, TP = 0.949, FP = 0.074, ROC = 0.937



K-nearest neighbors: Accuracy = 0.941518, TP = 0.942, FP = 0.092, ROC = 0.927

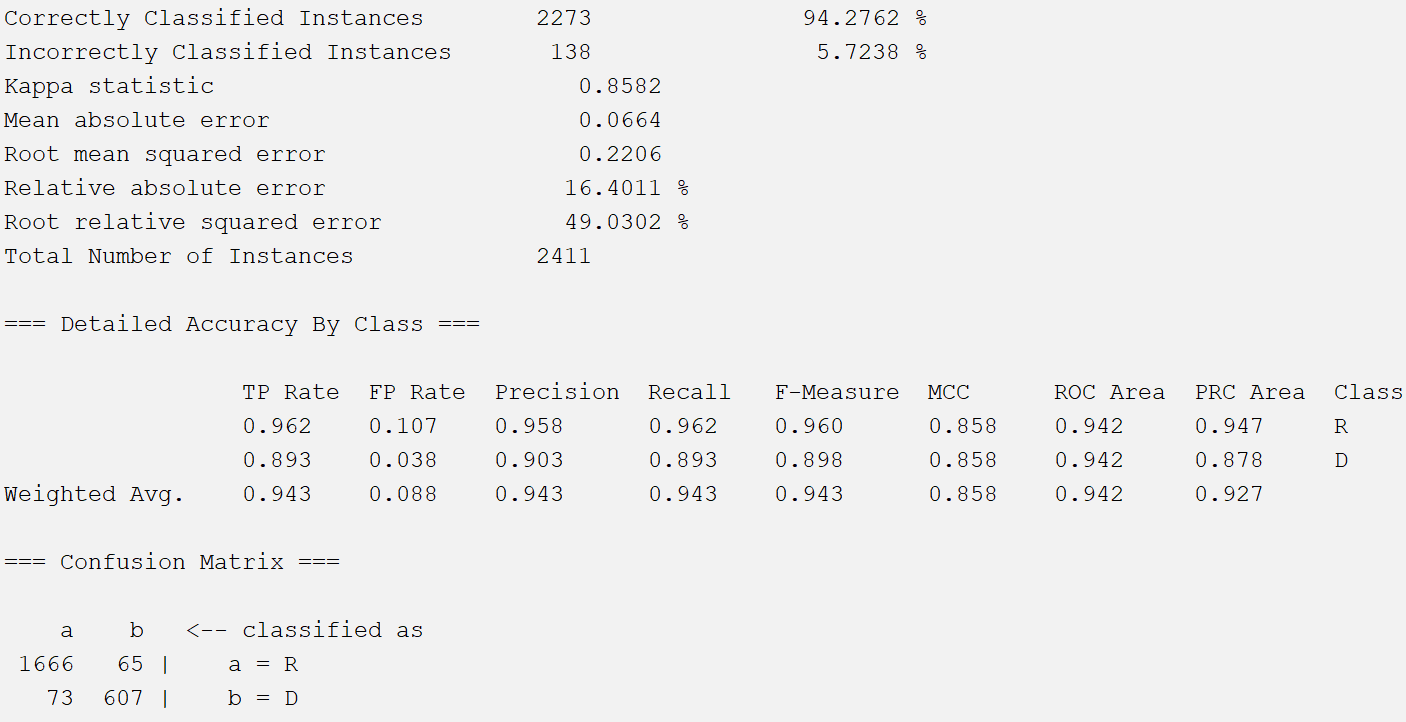


Naïve Bayes: Accuracy = 0.92866, TP = 0.929, FP = 0.046, ROC = 0.985

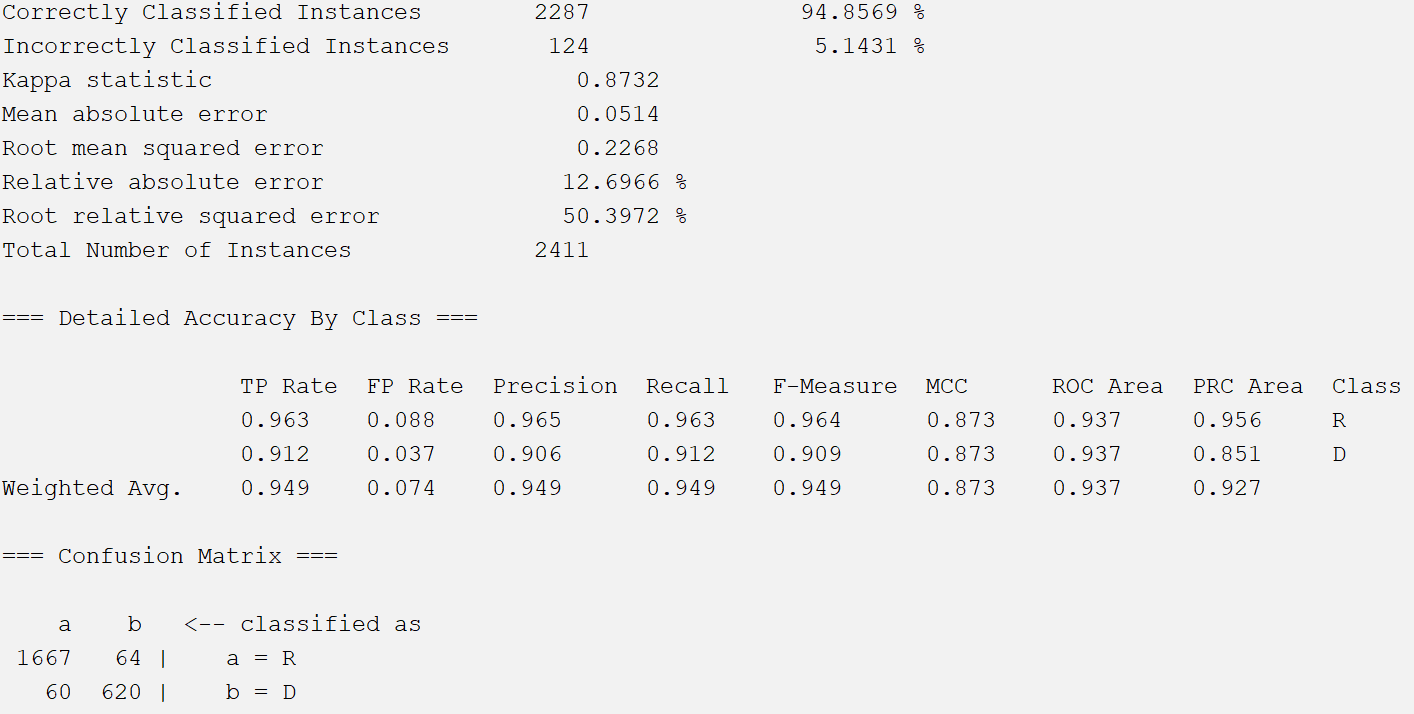


***ReliefF Dataset:***

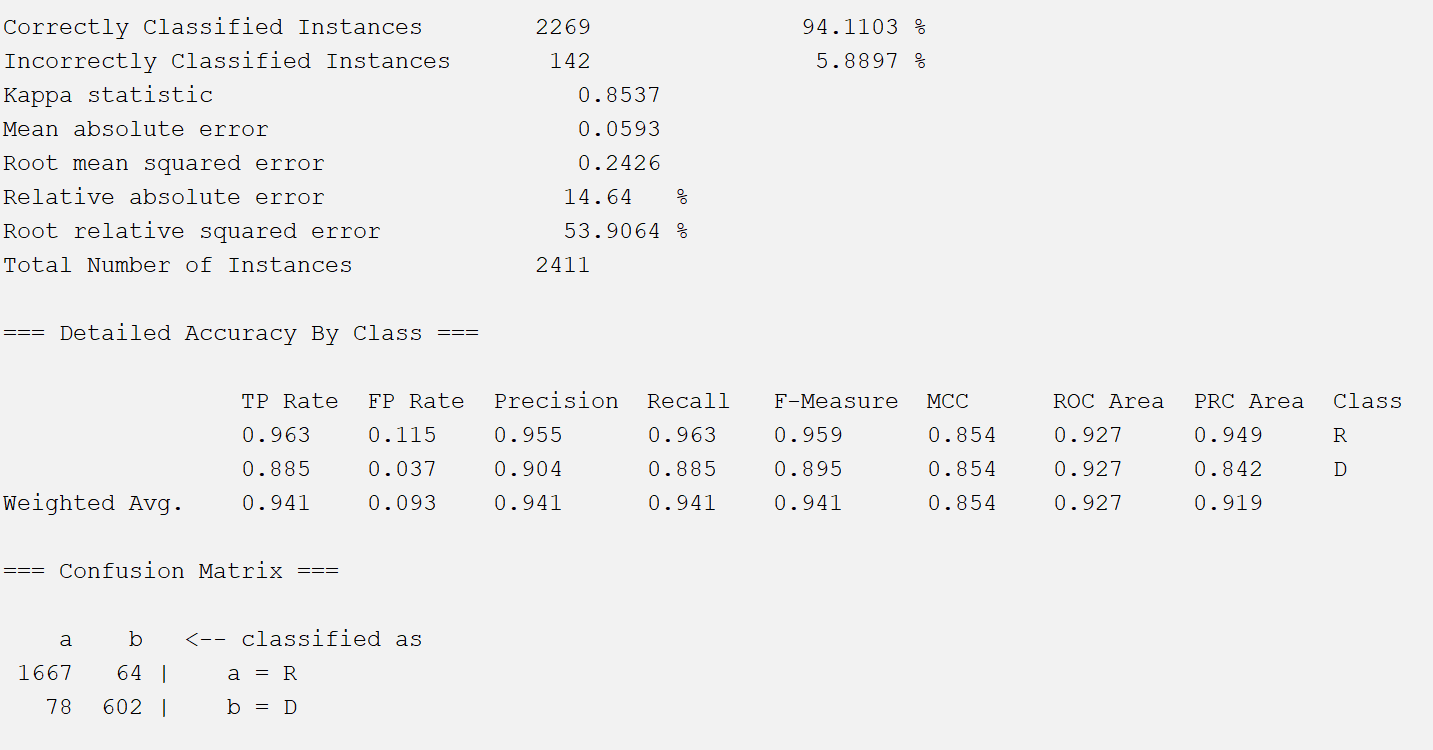
J48 decision tree: Accuracy = 0.942762, TP = 0.943, FP = 0.088, ROC = 0.942

**

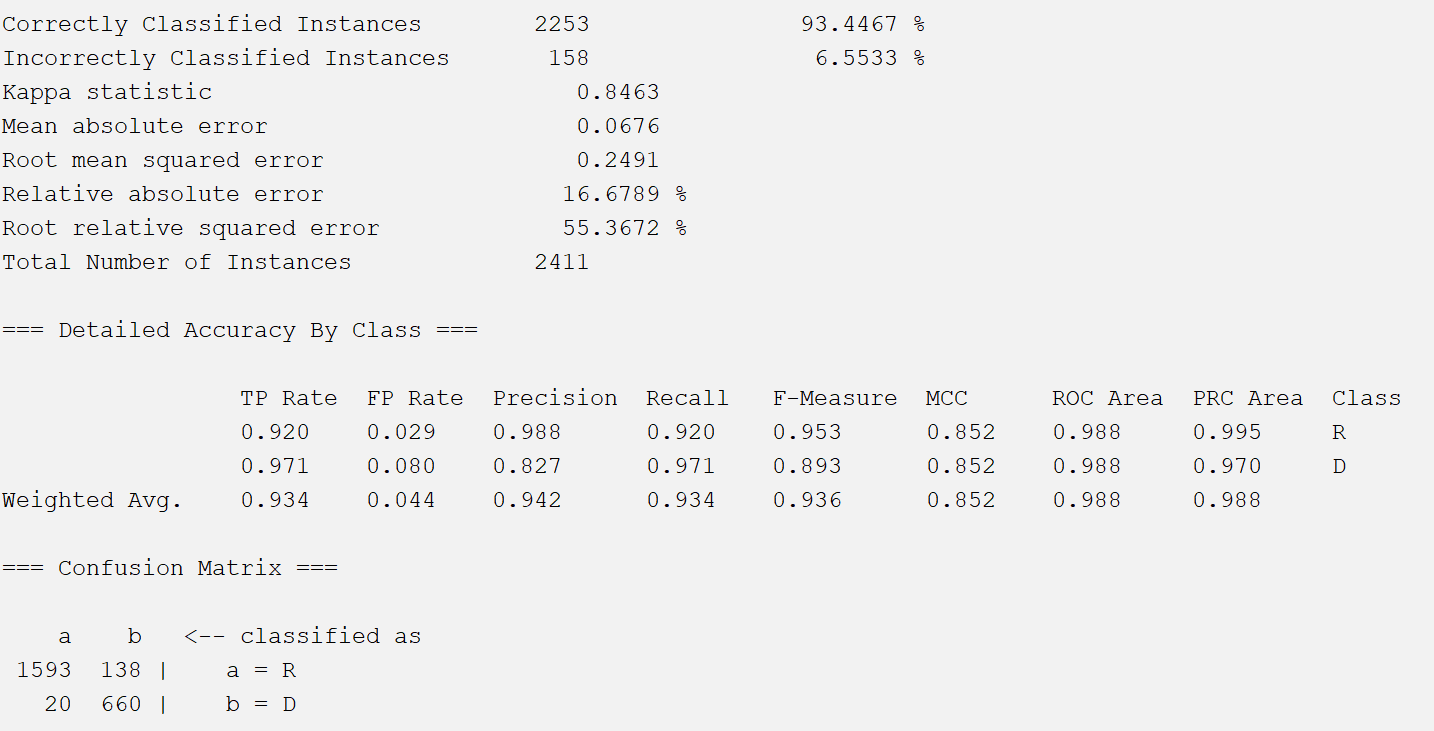
OneRClassifier: Accuracy = 0.948569, TP = 0.949, FP = 0.074, ROC = 0.937

**

K-nearest neighbors: Accuracy = 0.941103, TP = 0.941, FP = 0.093, ROC = 0.927

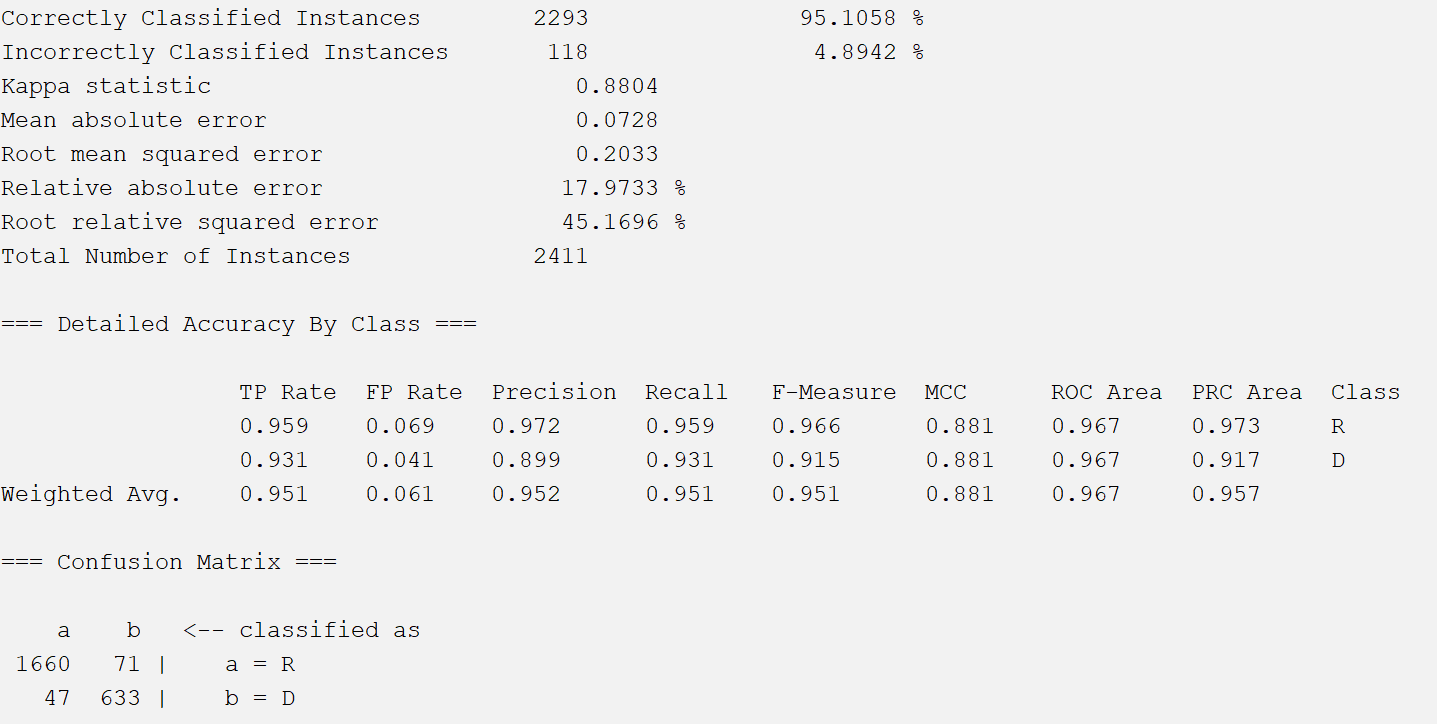


Naïve Bayes: Accuracy = 0.934467, TP = 0.934, FP = 0.044, ROC = 0.988

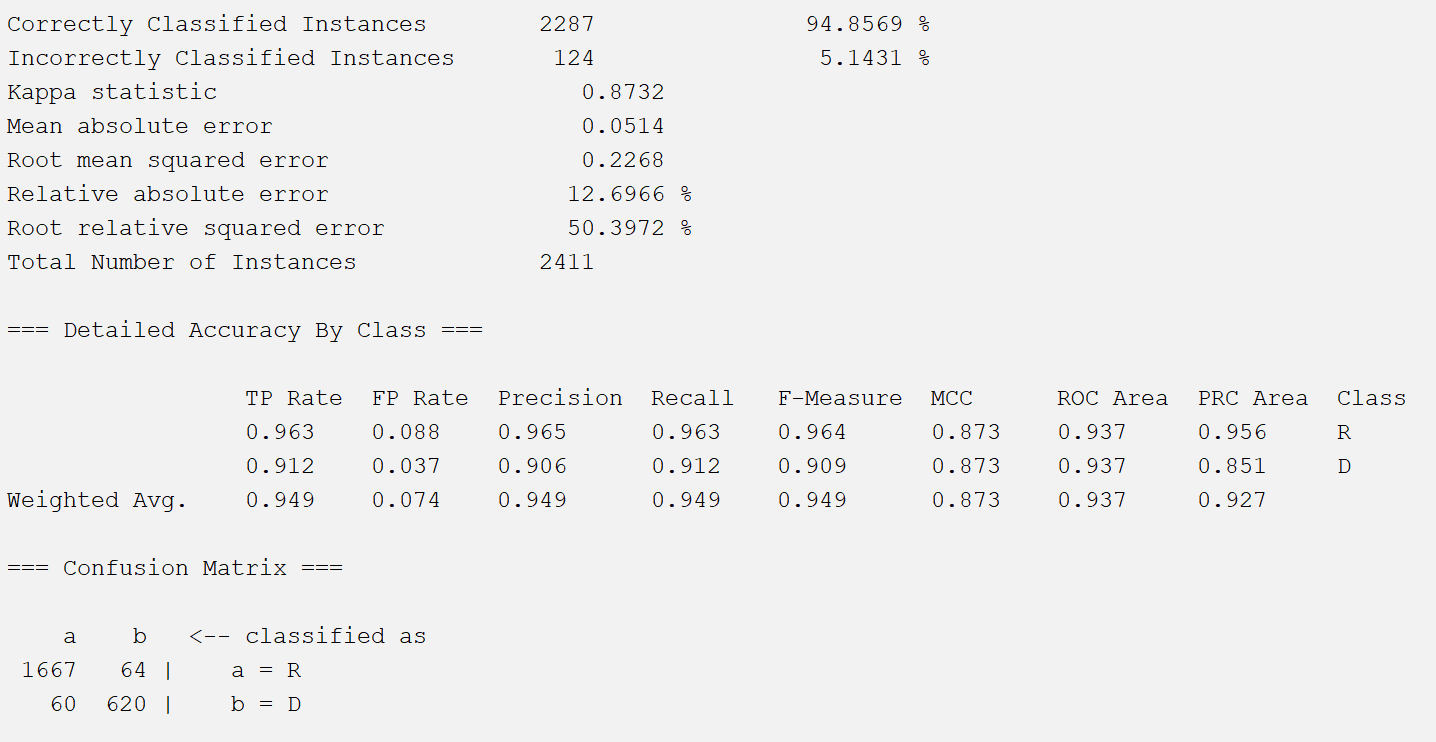


***InfoGain Dataset:***

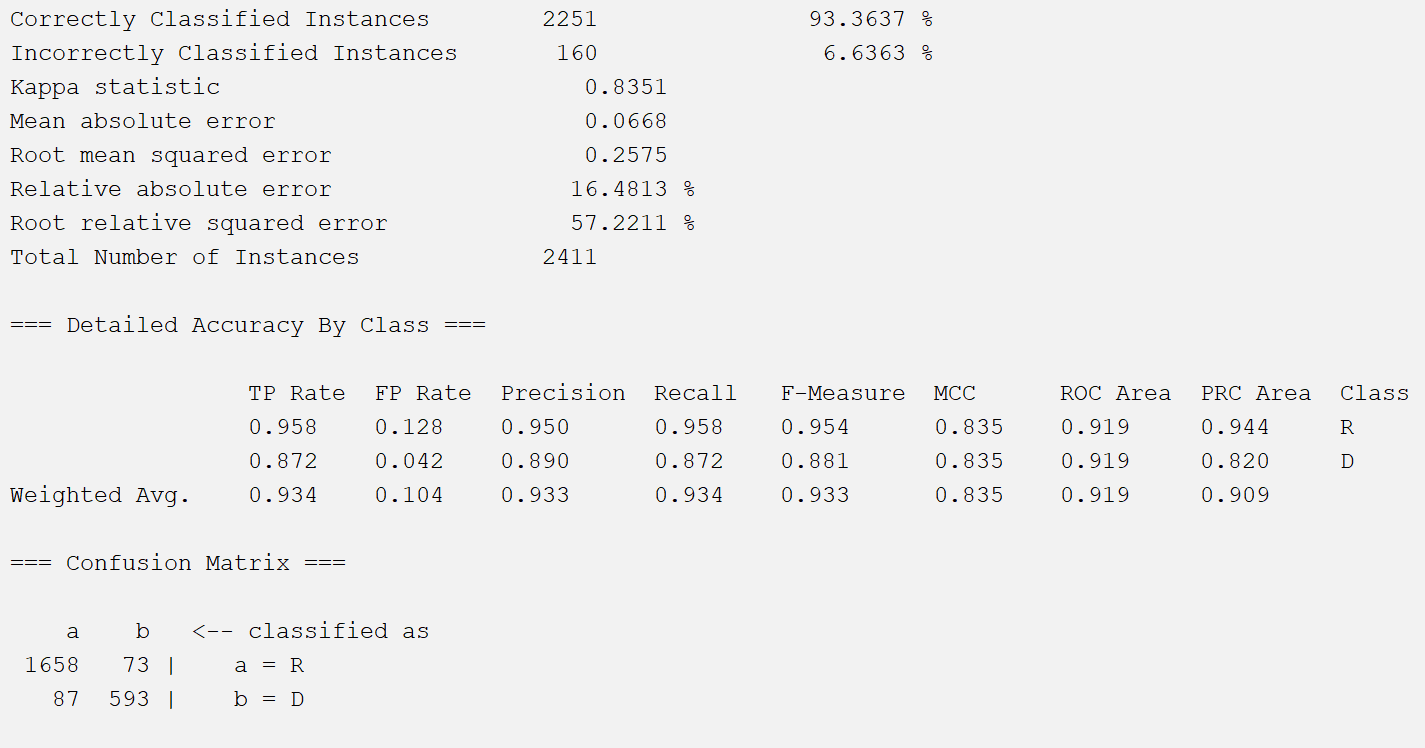
J48 decision tree: Accuracy = 0.951058, TP = 0.951, FP = 0.061, ROC = 0.967



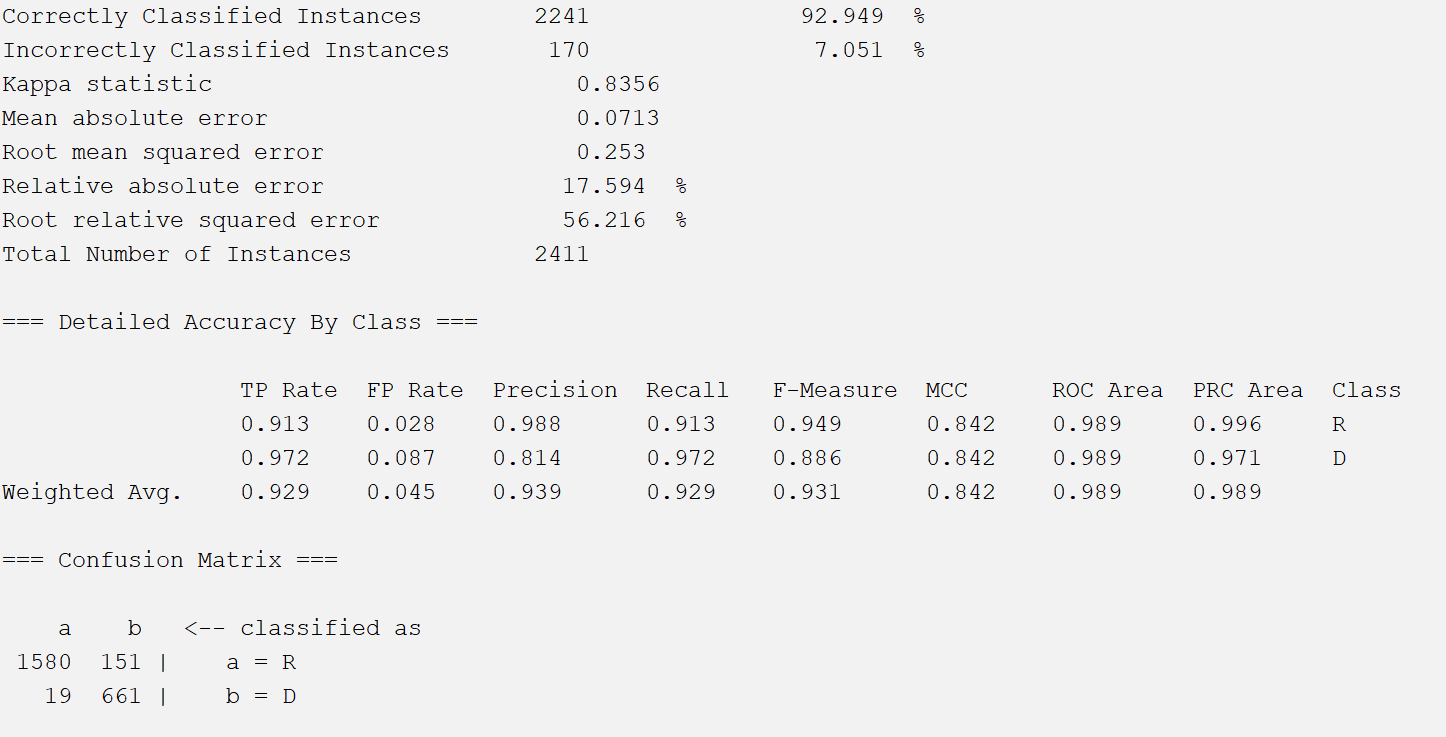
OneRClassifier: Accuracy = 0.948569, TP = 0.949, FP = 0.074, ROC = 0.937



K-nearest neighbors: Accuracy = 0.933637, TP = 0.934, FP = 0.104, ROC = 0.919

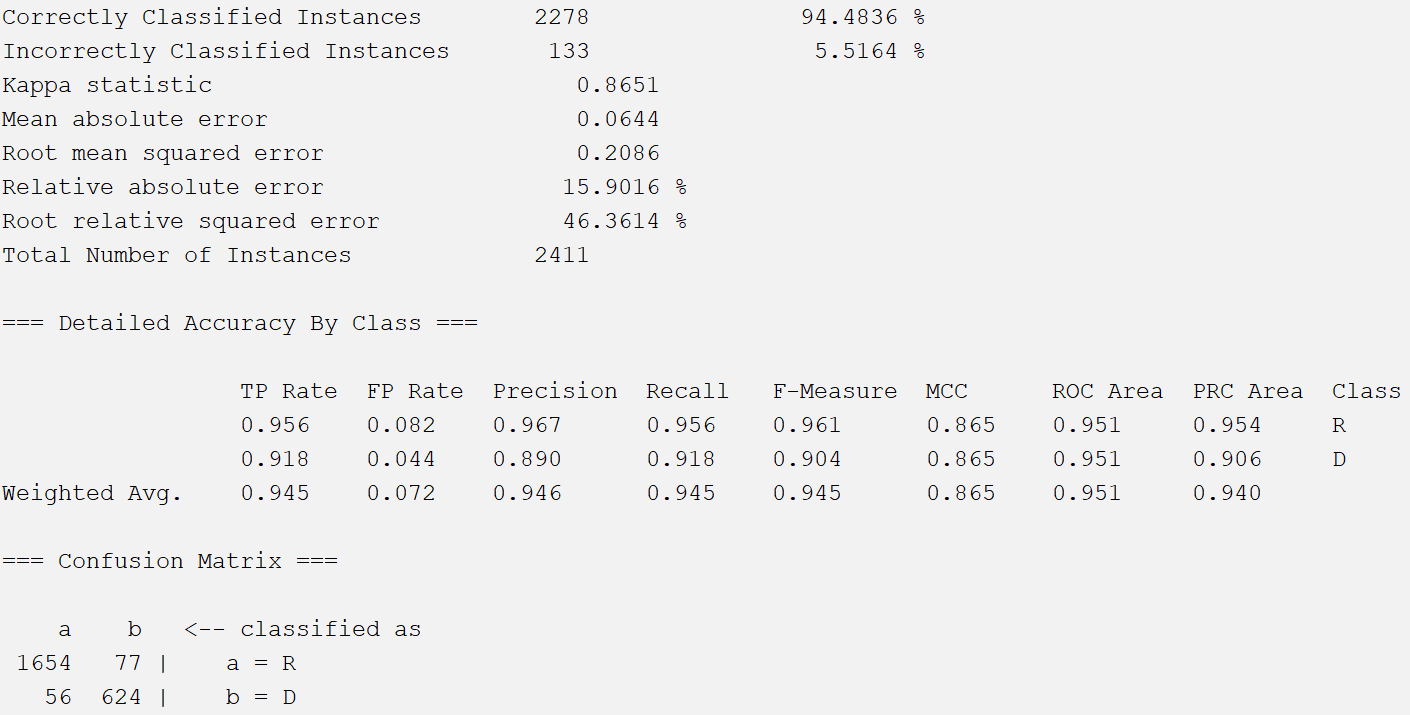


Naïve Bayes: Accuracy = 0.92949, TP = 0.929, FP = 0.045, ROC = 0.989

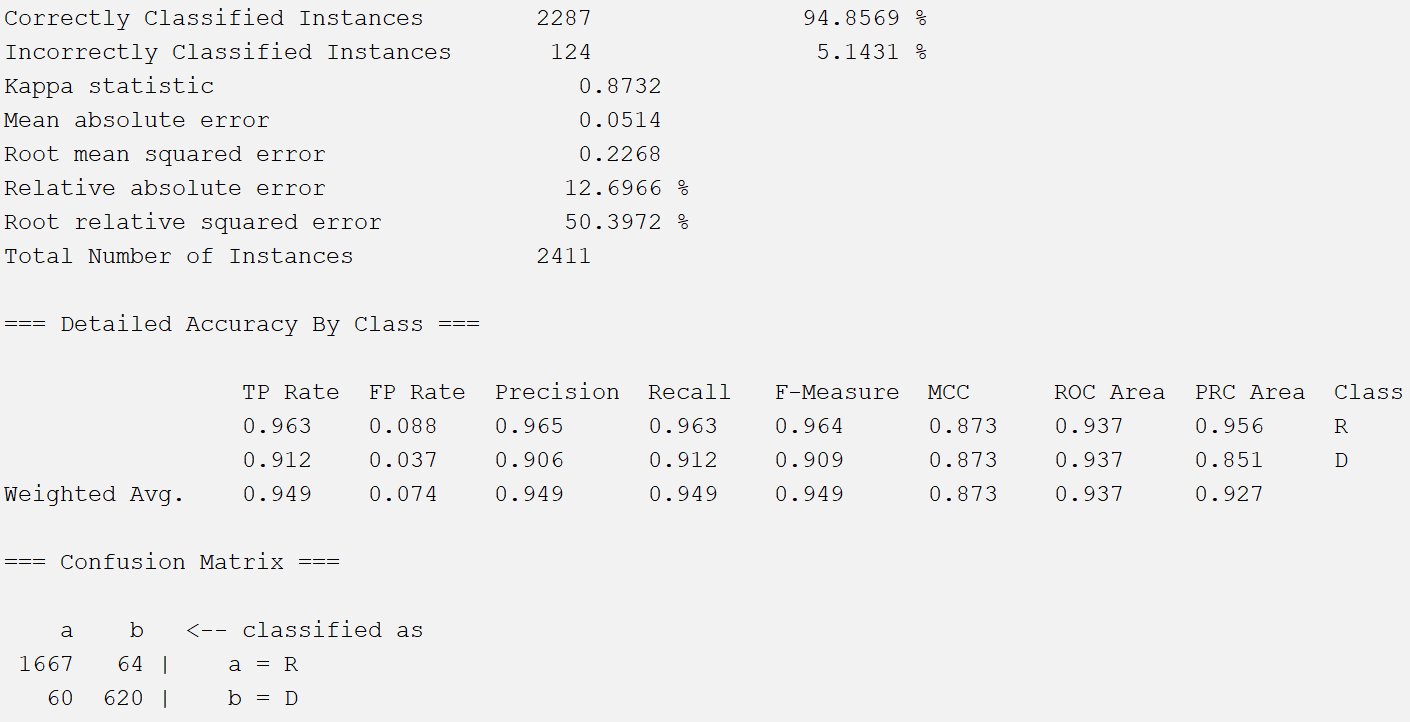


***Correlation Dataset:***

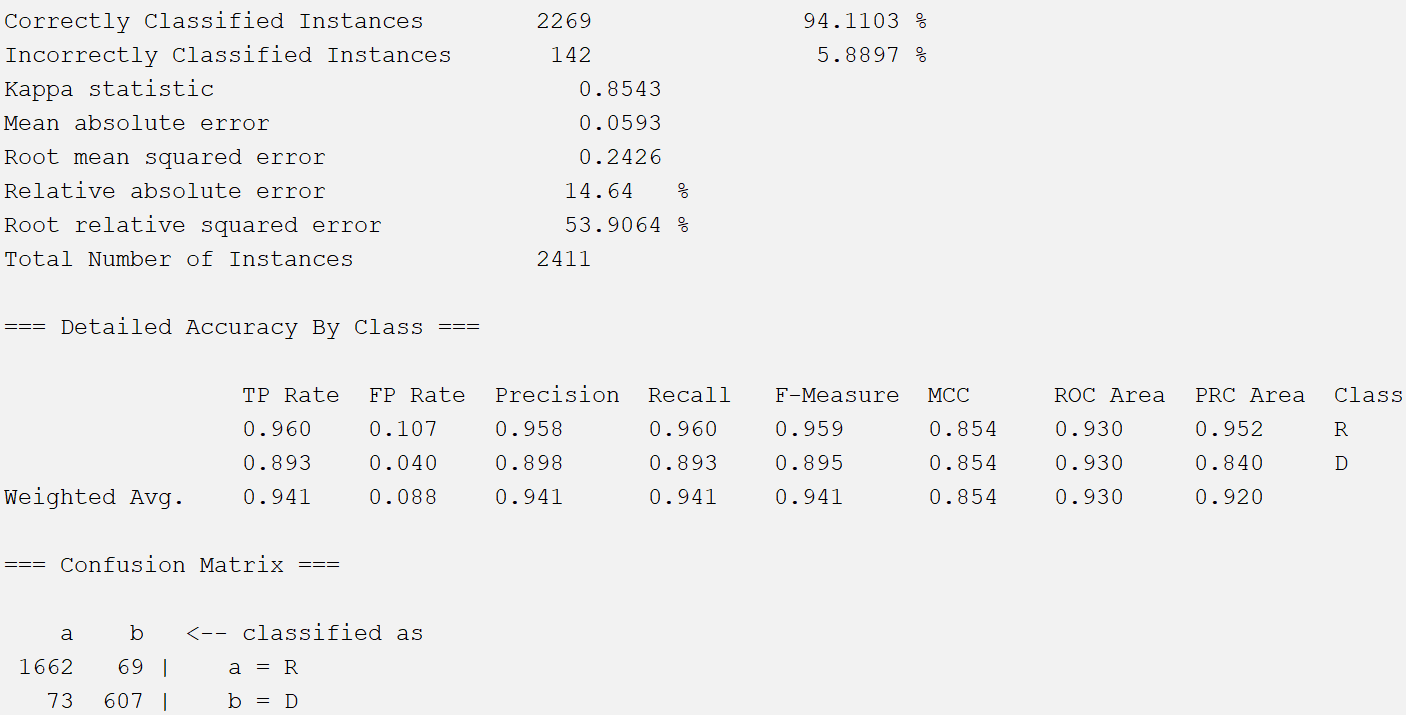
J48 decision trees: Accuracy = 0.944836, TP = 0.945, FP = 0.072, ROC = 0.951



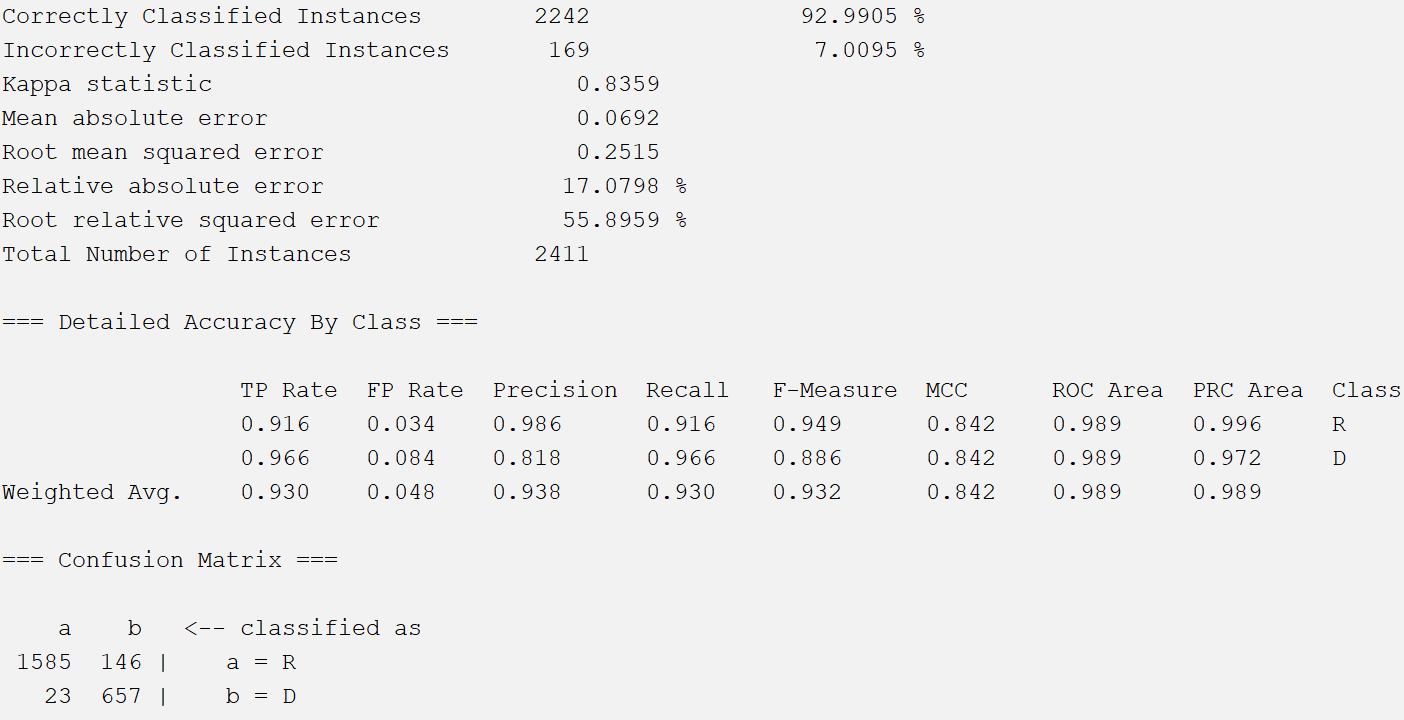
OneRClassifier: Accuracy = 0.948569, TP = 0.949, FP = 0.074, ROC = 0.937



K-nearest neighbors: Accuracy = 0.941103, TP = 0.941, FP = 0.088, ROC = 0.930

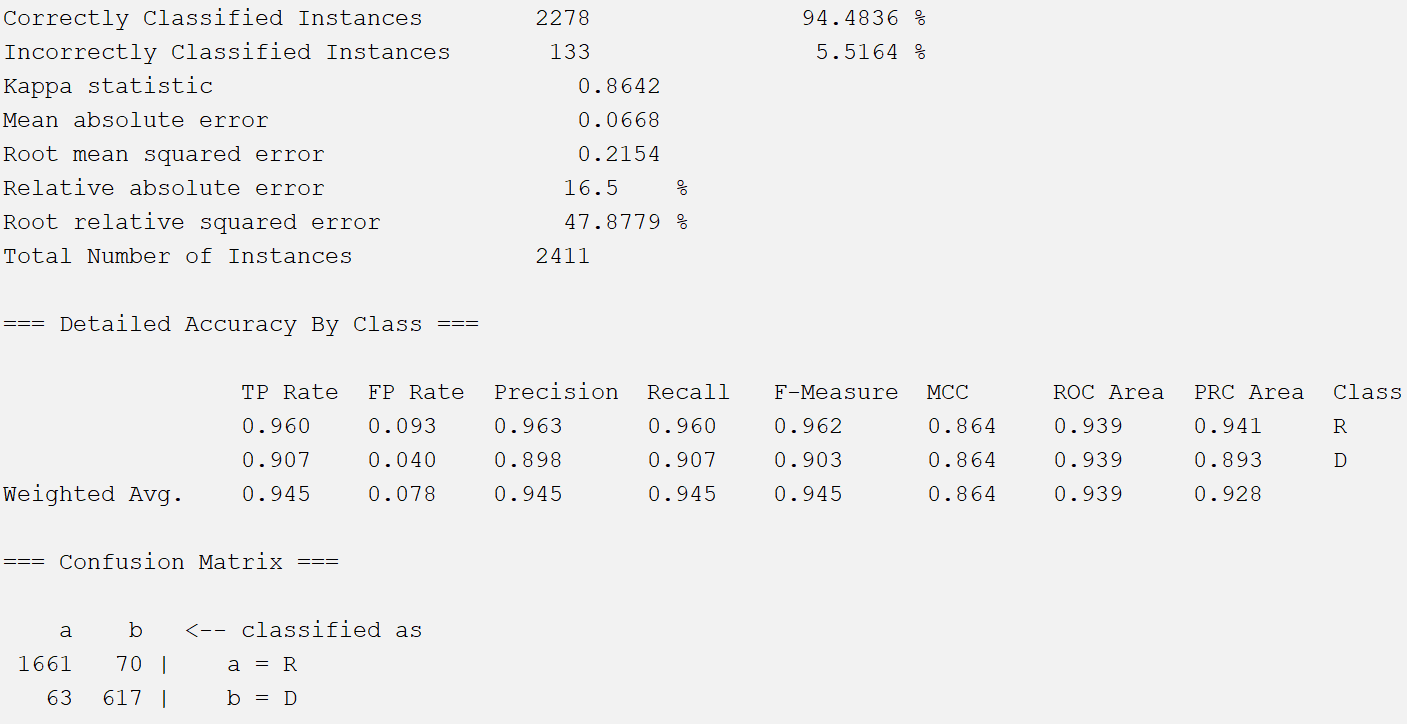


Naïve Bayes: Accuracy = 0.929905, TP = 0.930, FP = 0.048, ROC = 0.989

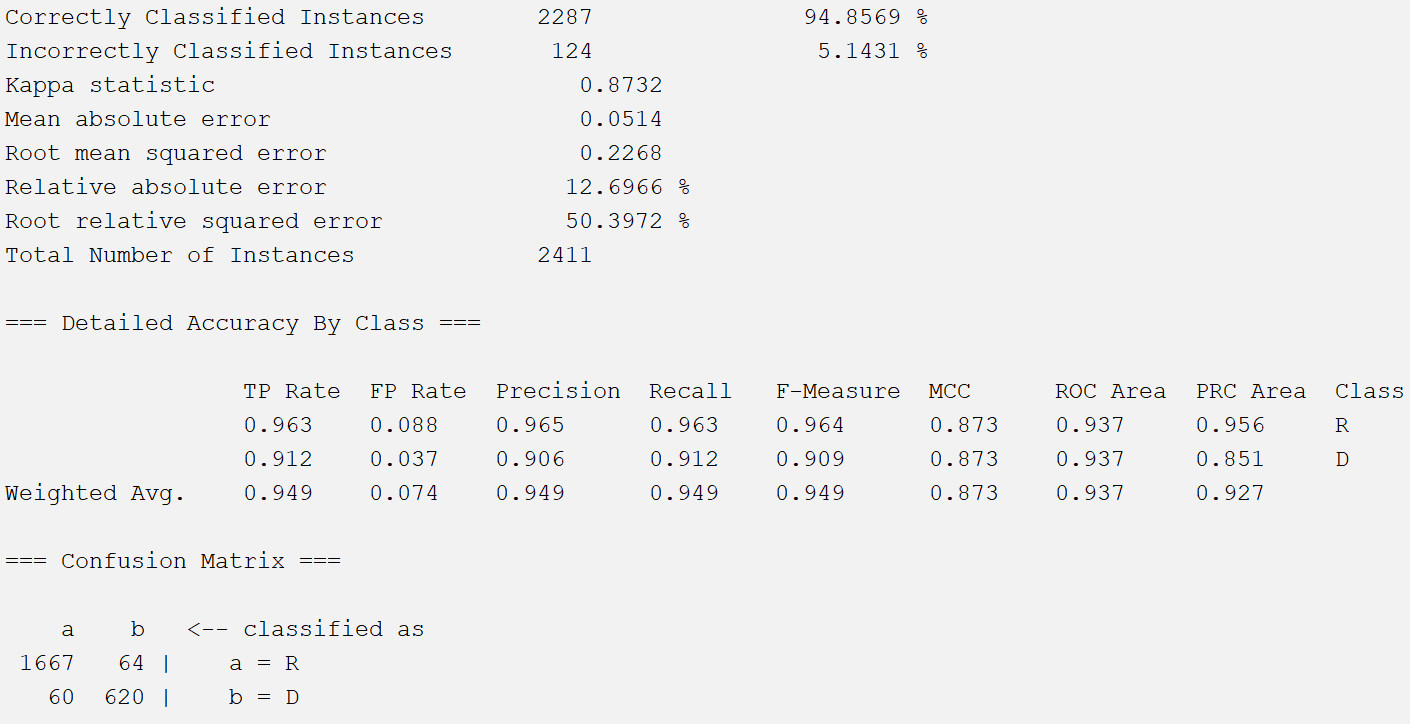


***Dataset with Our Chosen Attributes:***

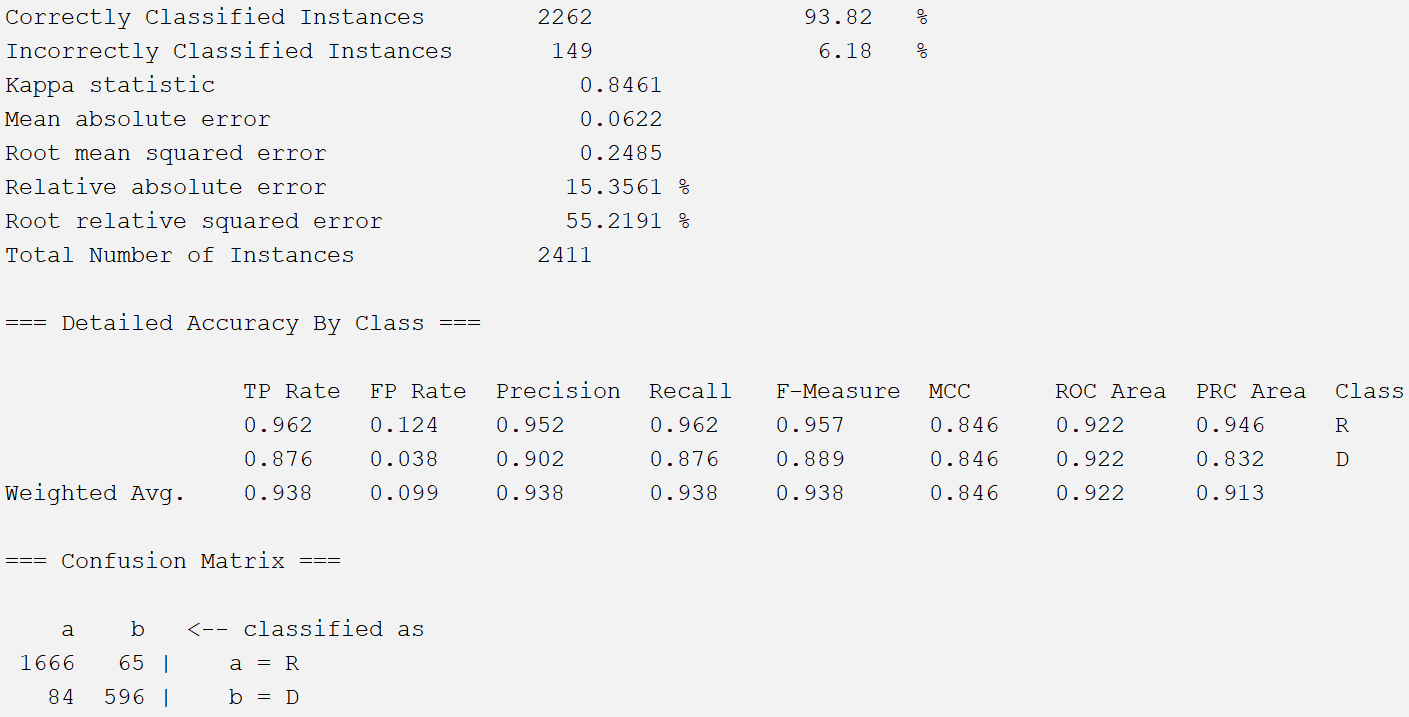
J48 decision tree: Accuracy = 0.944836, TP = 0.945, FP = 0.078, ROC = 0.939



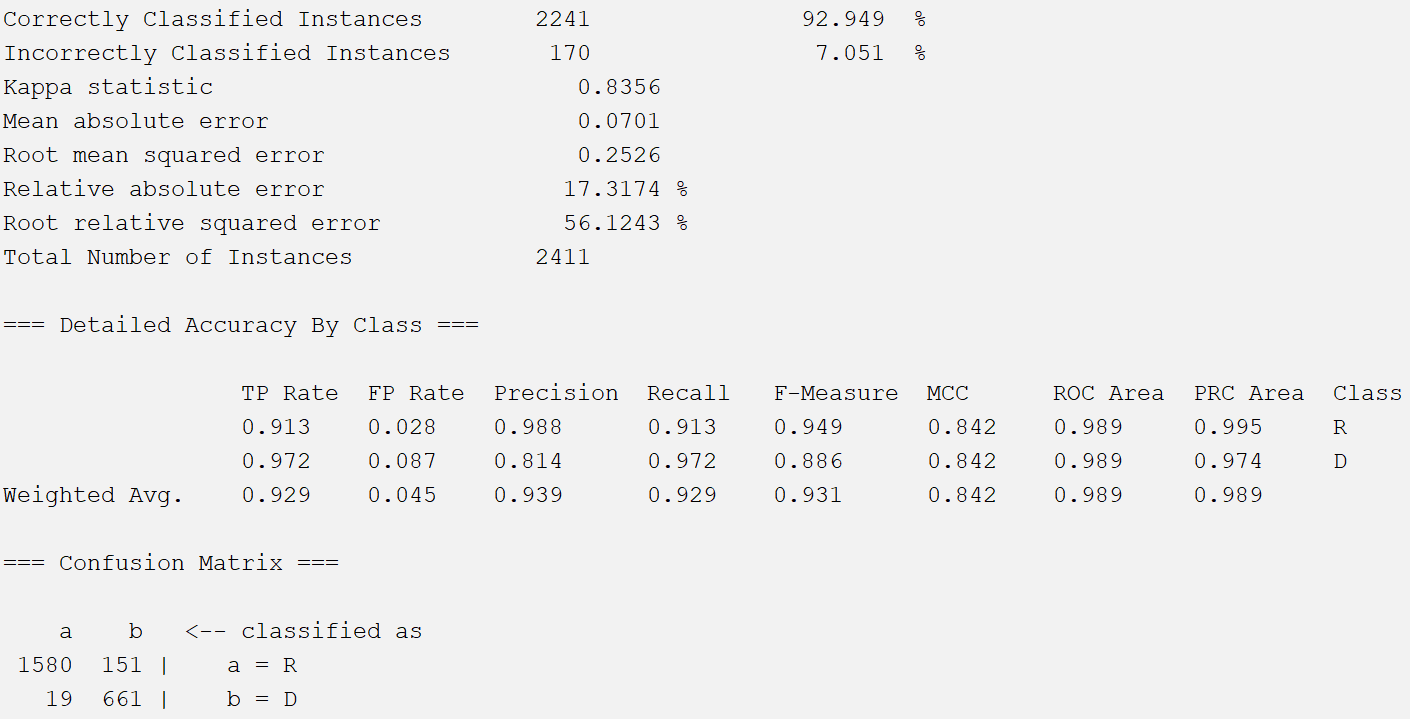
OneRClassifier: Accuracy = 0.948569, TP = 0.949, FP = 0.074, ROC = 0.937



K-nearest neighbors: Accuracy = 0.9382, TP = 0.938, FP = 0.099, ROC = 0.922



Naïve Bayes: Accuracy = 0.92949, TP = 0.929, FP = 0.045, ROC = 0.989



The first thing to note is that the OneRClassifier for each dataset achieved the same results. This is likely because each dataset includes all of the previous election results and thus, the OneRClassifier picked the same attribute for classification for each dataset (probably either the 2018 US Senate election or the 2016 US Presidential election).

As our model simply predicts whether a precinct is Democratic or Republican, false positives and false negatives do not have any significant meaning like they may in a field like medicine. In addition, except for Naive Bayes, which classified a large number of Republican precincts as Democratic, none of the models had a particularly high number of false positives or false negatives. Thus, we used accuracy as the primary metric to assess each model. For this reason, the best model is the one using OneR for attribute selection and the J48 decision tree for classification. This combination achieved an accuracy of 95.4791%, almost four tenths of a percent higher than the second best model, the InfoGain/J48 combination, which achieved an accuracy of 95.1048%. Beyond this, each of the 4 OneR classifier models achieved a reasonably high accuracy of 94.8569%, while the K-nearest neighbor models sat in the 93 to low 94 range for accuracy, and Naïve Bayes brought up the rear with accuracies in the 92s and low 93s. From an attribute selection standpoint, OneR attribute selector was the best, although none was consistently superior.

**Discussion and Conclusion**

The first and most obvious takeaway is that a combination of OneR attribute selector with a J48 decision tree produced the best results. However, there are a number of other important conclusions we can draw from our report.

First, it is evident that past electoral results are superb predictors of future results. The 9 past elections were ranked as the top attributes by each of our four attribute selection techniques, indicating that these results are better predictors of future results than any other data we tested.

In addition, we noticed that the most predictive racial attributes were the White and Black populations expressed as a percentage of the population. While the proportions of other racial/ethnic groups had some correlation with electoral results (Hispanics, for example), the attribute selection techniques consistently picked measures of both the White and Black populations.

We did not, however, observe a meaningful difference between Voting Age and Total populations for each group, indicating that these metrics can be used interchangeably in future models. There were some other surprises as well: the land area of a precinct ended up being a reasonably good predictor of partisan lean, while the total housing was not. Future investigators can build upon these findings when creating models.

Each of the models ran quite quickly, with none exceeding 0.1 seconds in runtime. In the future, this study could be improved upon by considering more classifiers and more attribute selection methods. Furthermore, a superior K-fold value could be discovered to potentially improve accuracy.

**Division of Responsibilities between Group Members**

Deven

* Found datasets on Redistricting Data Hub and interpreted attributes
* Created script for train/test/validation split
* Deleted empty columns and precincts with missing values/tied precincts
* Performed min/max and decimal scaling normalization to make precinct dataset completely normalized

Vibby

* Ran attribute selection algorithms in WEKA
* Created new datasets for each attribute selection technique by deleting unnecessary columns
* Performed the train/test/validation split for each dataset (although we ended up using k-fold cross-validation instead)
* Built and tested all 20 models using WEKA

Both

* Worked together to write this lab report
* Discussed, both at home and in class, each person’s personal responsibilities to ensure that we understood all parts of the project

**Links to Sources**

1. <https://redistrictingdatahub.org/dataset/virginia-block-county-and-vtd-pl-94-171-2020-official>
2. <https://www.virginiaredistricting.org/PageReader.aspx?page=2020DataDownload>
3. <https://docs.google.com/presentation/d/1C_lwPZ02HMdpuLqNiqIkTB4prtIW79b7LzonvXq731Y/edit#slide=id.g2fea41556c7_1_0>
4. <https://docs.google.com/presentation/d/1dsZEHmHqs0FmfawpmHd8xc15hyQLaaWrE6mB61p7ilA/edit#slide=id.g2f4ed5cce61_0_0>