Machine Learning 1 - Quarter 1 Project Final Report

Group members: Isabella Zhu, Lilian Zhu

Date: 10/03/2022

Period: 5

Please see the Project Description: <https://docs.google.com/document/d/1SEWnZweAofhKJYfGAkBncH_P_LNrQfggb-N6pTRsXgA/edit>

So far, you have done Proposal, Intermediate Report. You will submit the remaining items for the **final report.**

Please create a google drive folder to place your raw data, preprocessed data, training and test data as well as your final report and your presentation slides. You may arrange your final report in any way you want as long as you include all the information mentioned in the Project Description. However, to give you an idea, you may include a table of context like the one I copied below:

**Table of Contents**

Part 1 – Statement/Project Goal 3

Part 2 – Description of Dataset 4

Part 3 – Pre-Processing 5

Part 4 – Attribute Selection Algorithms & Model Classifiers Used 8

Part 5 – Results and Analysis 12

Part 6 – Conclusion/How to Reproduce Our Model 16

Part 7 - Team Members and Tasks Performed 18

Part 8 – Appendix and Sources 19

We will do the presentations during the week of 10/11/2022-10/26/2022. However, you need to submit your **slides** along with your final project submission Oct 11, 2022.

As the grades will be included in Quarter 1, there will be no time extensions on either the project or the presentations.

Please share your google drive folder with me: [syilmaz@fcpsschools.net](mailto:syilmaz@fcpsschools.net)

Copy the link to your google drive folder below. This will count as the submission of your final project and your presentation slides. **Link:** [**https://drive.google.com/drive/folders/1zxALY9xXp2kdhhrMK7mi9MlHaC6tBSYq**](https://drive.google.com/drive/folders/1zxALY9xXp2kdhhrMK7mi9MlHaC6tBSYq)

**Predicting COVID Deaths Using Machine Learning**

Machine Learning Quarter 1 Project Final Report

Isabella Zhu and Lilian Zhu

**Project Goal**

Our goal is to predict whether a patient will die from COVID-19 based on the attributes. This includes information about the patient’s demographic, such as their age, sex, race, ethnicity, and location. It also includes details about their exposure to COVID, such as how they were exposed to COVID, how long it took symptoms to manifest, and how their COVID was diagnosed. Furthermore, the data includes attributes for each patient’s current conditions, such as whether they were admitted to the hospital or an emergency room.

By combining data about these attributes, we hope to predict whether any given patient will die from COVID-19. There are many practical applications of this. Firstly, this would be useful for hospitals, which need to identify high-risk patients in order to focus intensive care on those most likely to die. Secondly, it helps keep the government and politicians accountable: If, after a new government initiative, there is an uptick in the different attribute values that correspond to high death rates, then policymakers will have forewarning to roll back the initiative before lasting harm is done. On the other hand, if a new government initiative is helpful in suppressing a data point that our model shows is correlated to high death rates, then knowing so helps roll out the program more quickly and prevent unnecessary deaths. Thirdly, this program predicts the lethality rate of COVID *taking demographics into account*. Thus, it can be used to predict COVID deaths as a whole, but also the lethality of COVID in a specific geographical area, given common census data such age, sex, race, and ethnicity. When apportioning COVID relief funds, naturally national and state governments wish to place emphasis on the areas that will be hardest-hit. Our program has the potential to make sure the money goes where it’s needed, *before* it’s needed.

**Dataset Description**

**Source:** <https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data-with-Ge/n8mc-b4w4>

This dataset is from the CDC and contains COVID case surveillance data. Each instance is a COVID case. There are 19 attributes (less than 20, but this was approved), and 82,313,422 instances. Death\_yn is the class attribute.

The attributes are:

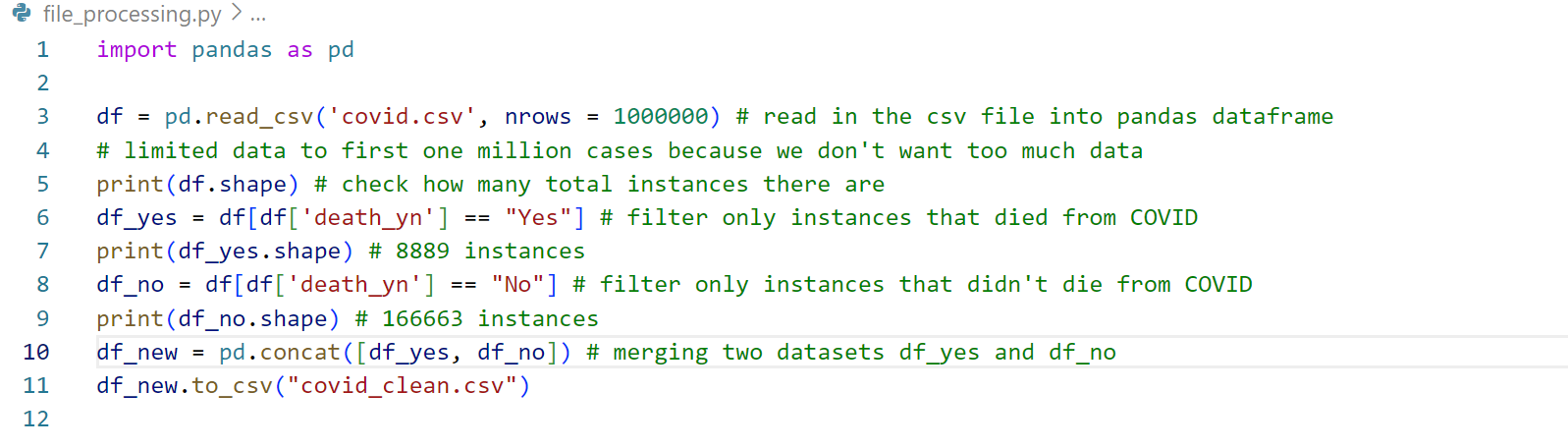
* case\_month: the earlier of month the Clinical Date (date related to the illness or specimen collection) or the date received by CDC
* res\_state: state of residence
* state\_fips\_code: state FIPS code
* res\_county: county of residence
* county\_fips\_code: county FIPS code
* age\_group: age group (0-17 years, 18-49 years, 50-64 years, 65+ years)
* sex: female, male, other
* race: American Indian/Alaska Native, Asian, Black, Multiple/Other, Native Hawaiian/Other Pacific Islander, White
* ethnicity: Hispanic, Non-Hispanic
* case-positive\_specimen\_interval: weeks between earliest date and date of first positive specimen collection
* case\_onset\_interval: weeks between earliest date and date of symptom onset
* process: the process in which the case was first identified (clinical evaluation, routine surveillance, contact tracing, multiple, other)
* exposure\_yn: did the patient have any known exposures, such as travel, work, etc.
* current\_status: current status of person (lab-confirmed case, probable case)
* symptom\_status: symptom status of person (asymptomatic, symptomatic, unknown)
* hosp\_yn: was the patient hospitalized?
* icu\_yn: was the patient admitted to an intensive care unit?
* underlying\_conditions\_yn: did the patient have underlying conditions?
* death\_yn: did the patient die?

**Preprocessing**

Our data contains 82,313,422 instances; unfortunately, the size of the dataset means that attempting to do any data preprocessing with the entire dataset crashes our computers. From a cursory glance, the class distribution for this dataset is not uniform; There is not a 50/50 split between people who died and people who did not die from COVID. Unfortunately, we cannot list the exact ratio of people who died/people who didn’t die since both WEKA and Python crash from the size of the dataset (From a quick Google search of COVID lethality rates, however, it should be somewhere around 0.01). Our original plan to cut down on the size of the dataset was to split the dataset based on class and to take samples from the dataset such that class distributions are maintained. Unfortunately, this also crashed our computers. We were also unable to find a smaller version of our current dataset. Therefore, we took the first 1 million values from the dataset; Dr. Yilmaz indicated in class that this was fine for our project, since we lacked the computing power necessary to handle a larger quantity of data.

***Missing Values Handling***

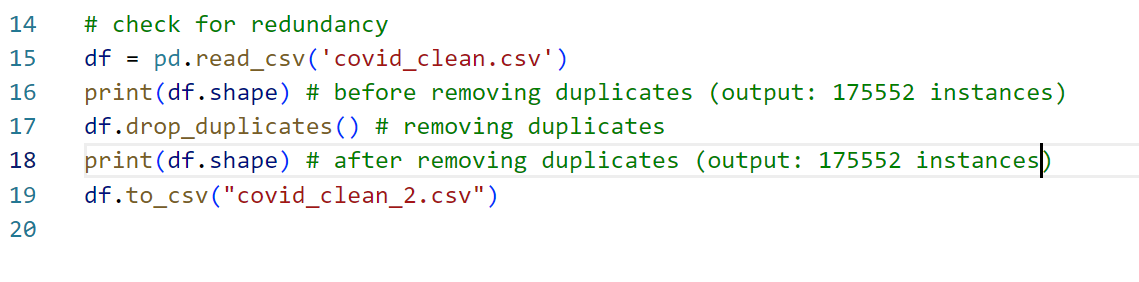
We removed all the instances with missing class labels, since they have no predictive power. The code for this is shown below.



We used the Pandas DataFrame structure to process data. We used a filtering method, shown on lines 6 and 8, that only keeps data with a definitive death\_yn class label (either yes or no). Then, we combined them into one dataset and saved it. The dataset covid\_clean.csv has no missing class labels.

***Redundancy***

We checked for redundancy (duplicate rows) using the Pandas DataFrame drop\_duplicates() method. The code for checking duplicates is shown below.



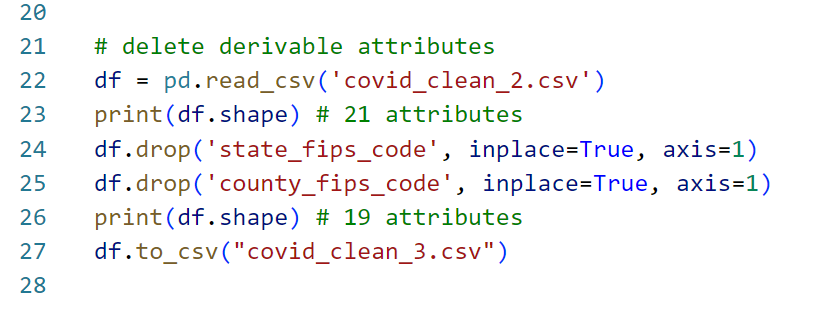
There were no duplicate rows because the shape of the dataframe did not change after applying the drop\_duplicates() method.

***Dimensionality Reduction***

To reduce the dimensionality of the data, we had two main steps. The first step was removing derivable attributes of the data, which we did in python. The second step was removing attributes that had a low correlation with the class label. We did this step in WEKA.

*Derivable Attributes*

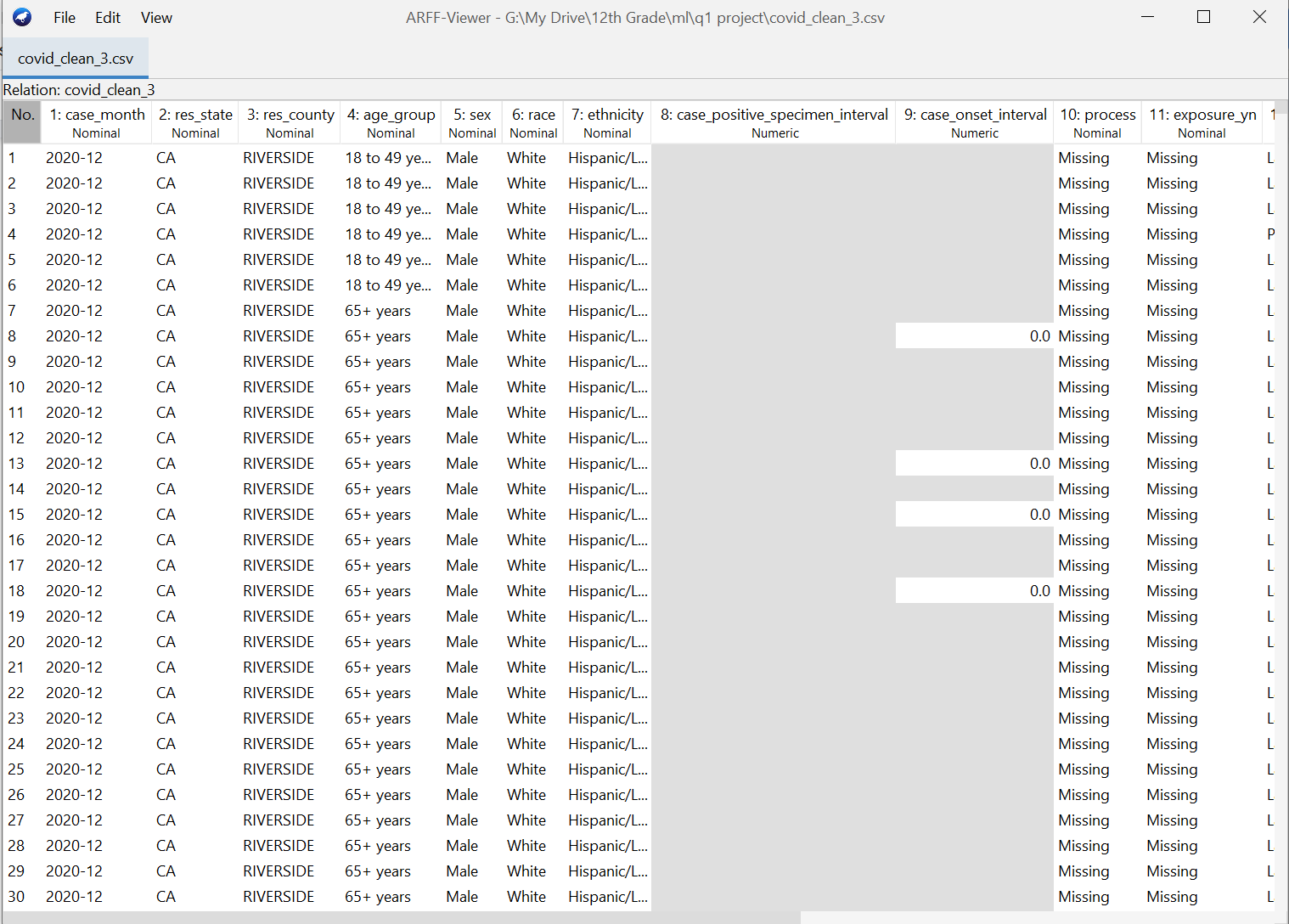
We deleted the attribute state\_fips\_code since it is a derivable attribute of res\_state, the state the patient lived in. (For example, if the patient lives in Alaska, their FIPS code is always 01.) We also deleted the attribute county\_fips\_code since it is a derivable attribute of res\_county, the county the patient lived in. (For example, if the patient lives in Autauga, their FIPS code is always 01001.) The code for this is shown below.



*Preparing for Processing in WEKA*

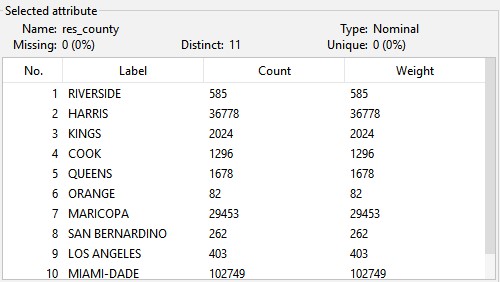
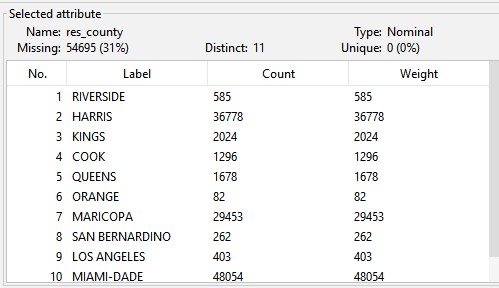
Next, we converted the csv file into an arff file so we could analyze correlation between attributes in WEKA. We converted the csv file to an arff file through WEKA’s ARFF-Viewer. This is accessible through Tools -> ARFF-Viewer.

In the ARFF-Viewer window, we opened the covid\_clean\_3.csv. Then, we saved it as covid\_clean\_3.arff. A screenshot of the csv file viewed through ARFF-Viewer is shown below.



*WEKA Processing*

We ran the dataset through the ReplaceMissingValues filter on WEKA.



pre-filter post-filter

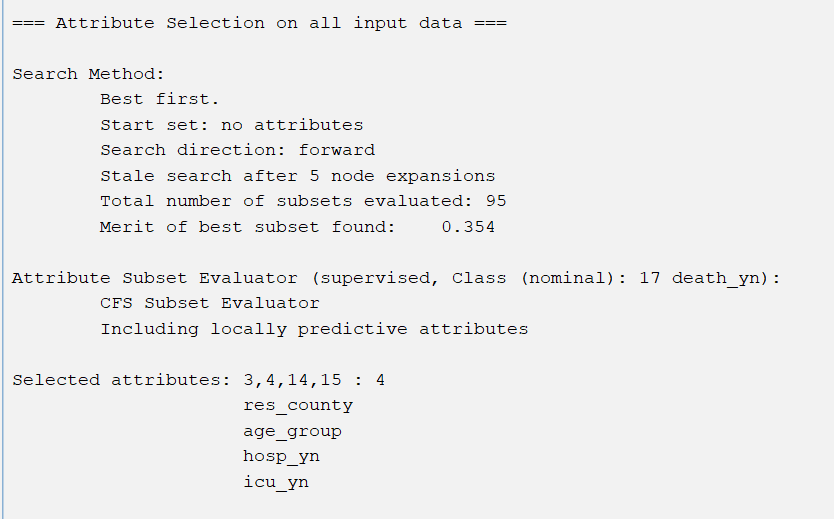
We replaced all the missing values. The result is saved as covid\_clean\_3\_no\_missing\_values.arff.

**Attribute Selection Algorithms and Model Classifiers Used**

**Attribute Selection Algorithms**

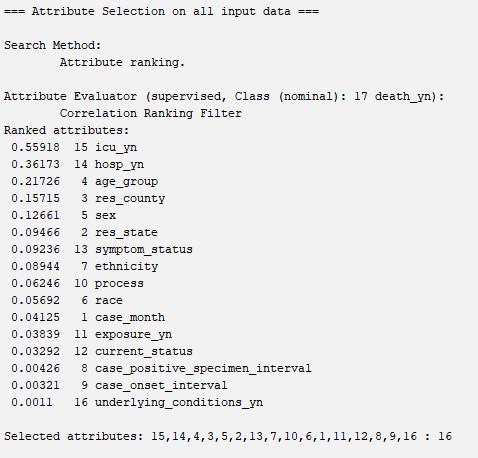
We used WEKA’s built in attribute selection algorithms, listed as “Attribute Evaluator” under the “Select Attributes” tab. For algorithms with a coefficient, we used a cutoff of 0.1 to select the significant attributes.

***Algorithm 1: CFS Subset Evaluation (Search Method: BestFirst)***

**

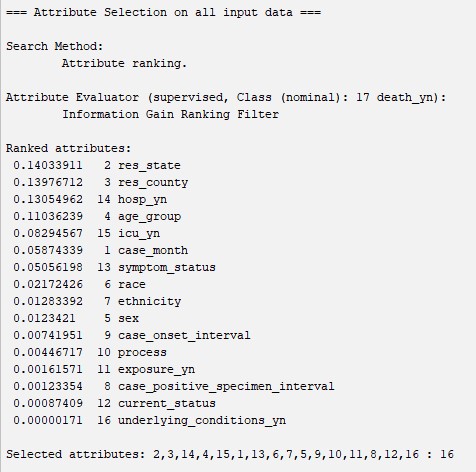
The attributes selected from this method are res\_county, age\_group, hosp\_yn, and icu\_yn.

***Algorithm 2: Correlation Attribute Evaluation (Search Method: Ranker)***



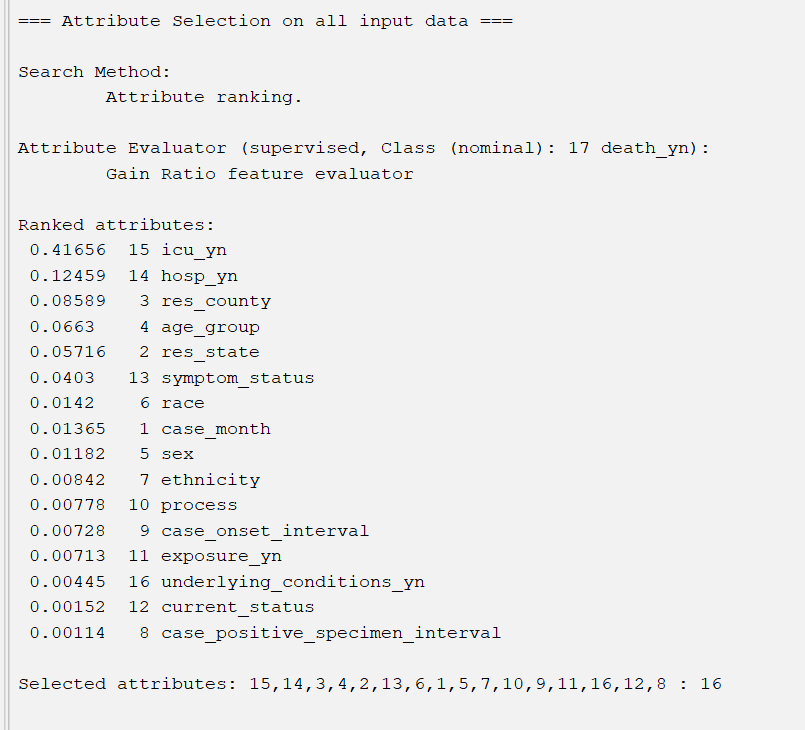
The attributes selected from this method are icu\_yn, hosp\_yn, age\_group, res\_county, and sex.

***Algorithm 3: Information Gain Attribute Evaluation (Search Method: Ranker)***



The attributes selected from this method are res\_state, res\_county, hosp\_yn, and age\_group.

***Algorithm 4: Gain Ratio Attribute Evaluation (Search method: Ranker)***

**

The attributes selected from this method are icu\_yn and hosp\_yn.

***Algorithm 5: Intuition***

We decided to choose these attributes for the following reasons.

* icu\_yn: If a patient is admitted into the ICU (which stands for Intensive Care Unit), then the patient must have a severe case of COVID and is thus more likely to die.
* hosp\_yn: Similar to the ICU reasoning, if a patient goes to the hospital for COVID, then the patient must have a severe case of COVID and thus is more likely to die.
* age\_group: It’s intuitive that older patients are more likely to die from COVID, since their immune systems are weaker than those of younger patients.
* exposure\_yn: If we know when the patient was exposed to COVID, then the exposure must have been significant. For example, working with someone who was diagnosed with COVID would likely result in a “yes” in this category, whereas passing someone on the street who had COVID would likely not have been noted. Therefore, people with a yes in this category are more likely to have imbibed a significant dose of COVID, and are thus more likely to develop a serious (and sometimes fatal) case of it.
* underlying\_conditions\_yn: People with underlying conditions are more likely to die from COVID because their immune system is overloaded, having to target both the COVID virus and their underlying condition.

**Training and Testing Sets**

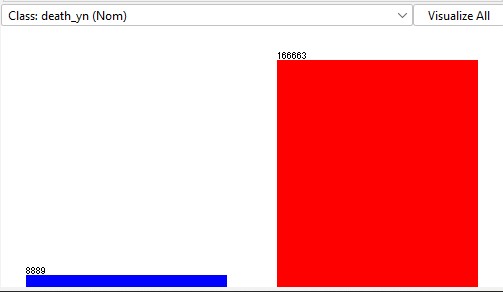
***Sampling***

Since our data is unevenly distributed, we used stratified sampling to guarantee a representative sample. We used WEKA to do this.

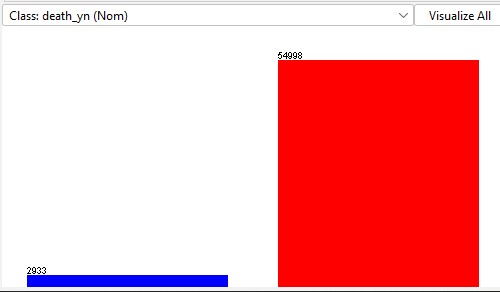
Split: 33% in testing, 67% in training

The data shown below is for the dataset chosen with the Correlation Attribute Evaluation method.

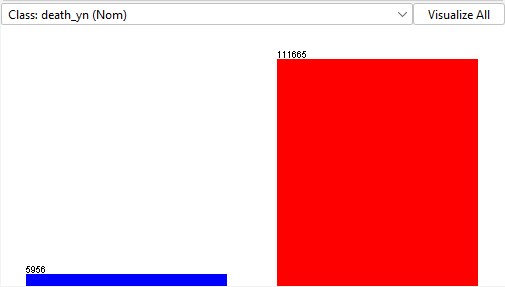
Below in order: Before split, testing, training



Before split: death to living ratio is 8869/166663 = 0.05321517073



Testing: death to living ratio is 2933/54998 = 0.05332921197



Training: death to living ratio is 5956/111665 = 0.05333810952

Since the class distributions for all three sets are equal, we can conclude that the test/training split is good and that both are representative of the original dataset. We also conducted the same analysis on the datasets chosen using the other four methods. The class distributions were maintained during the test/training split; thus, all test and training datasets are representative of their parent datasets.

**Model Classifiers**

We used the following model classifiers in WEKA under the “classify” tab in WEKA.

* Naive Bayes (bayes -> NaiveBayes)
  + Naive Bayes creates models based on each attribute’s correlation to the class attribute (in this case, how strongly each attribute correlates with patient deaths)
* J48 (trees -> J48)
  + J48 creates a model based on a decision tree.
* OneR (rules -> OneR)
  + OneR creates one predictive rule for each attribute. Thus, it does not take into account combinations of attributes.
* Decision Table (rules -> DecisionTable)
  + Similar to OneR, but there are multiple columns for different predictive rules, thus taking into account combinations of attributes.

The results from using these model classifiers are discussed in detail below.

**Results and Analysis**

The screenshots for testing results are included in the appendix.

***Accuracy Table***

These are the accuracy results. The top row is the attribute selection algorithm and the leftmost column is the classification model.

|  | CorrEval | InfoGain | CfsSubset | GainRatio | Intuition |
| --- | --- | --- | --- | --- | --- |
| Naive Bayes | 97.87% | 97.79% | 97.75% | 96.64% | 97.26% |
| J48 | 98.30% | 98.30% | 98.29% | 96.74% | 97.37% |
| OneR | 97.49% | 97.49% | 97.49% | 96.72% | 96.72% |
| De. Table | 98.29% | 98.29% | 98.28% | 96.73% | 97.36% |

Using J48 Decision Trees and CorrelationAttributeEvaluation or InfoGainAttributeEvaluation produced the best results in terms of accuracy.

***Confusion Matrix Table (Includes TP and FP)***

The format of our confusion matrices are as follows (this is consistent with WEKA):

|  | death\_yn = y (predicted) | death\_yn = n (predicted) |
| --- | --- | --- |
| death\_yn = y | a | b |
| death\_yn = n | c | d |

Thus, the true positive count is in the box marked “a” and the true positive rate (TP rate) is . The false positive count is in the box marked “c” and the false positive rate (FP rate) is .

These are the confusion matrix results. The top row is the attribute selection algorithm and the leftmost column is the classification model.

|  | | CorrEval | | InfoGain | | CfsSubset | | GainRatio | | Intuition | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Naive Bayes | | 2465 | 468 | 2324 | 609 | 2491 | 442 | 1204 | 1729 | 2101 | 832 |
| 764 | 54234 | 672 | 54325 | 863 | 54135 | 217 | 54781 | 753 | 54245 |
| J48 | | 2398 | 535 | 2398 | 535 | 2368 | 565 | 1104 | 1829 | 2073 | 860 |
| 452 | 54546 | 452 | 54546 | 426 | 54572 | 62 | 54936 | 666 | 54332 |
| OneR | | 1868 | 1065 | 1868 | 1065 | 1868 | 1065 | 1104 | 1829 | 1104 | 1829 |
| 389 | 54609 | 389 | 54609 | 389 | 54609 | 69 | 54929 | 69 | 54929 |
| De. Table | | 2393 | 540 | 2393 | 540 | 2374 | 559 | 1104 | 1829 | 2073 | 860 |
| 449 | 5459 | 449 | 54549 | 436 | 54562 | 63 | 54935 | 672 | 54326 |

***Area Under Receiver Operating Characteristic Curve***

These are the area under curve results. The higher the area under curve, the better the model is at distinguishing between positives and negatives. The top row is the attribute selection algorithm and the leftmost column is the classification model.

|  | CorrEval | InfoGain | CfsSubset | GainRatio | Intuition |
| --- | --- | --- | --- | --- | --- |
| Naive Bayes | 0.9879 | 0.9858 | 0.9867 | 0.9256 | 0.9782 |
| J48 | 0.9719 | 0.9732 | 0.9712 | 0.9255 | 0.9290 |
| OneR | 0.815 | 0.8149 | 0.8149 | 0.6876 | 0.6876 |
| De. Table | 0.9895 | 0.9898 | 0.9886 | 0.9258 | 0.979 |

***Analysis***

We analyzed our results through the following metrics. However, since our data’s class label is very unevenly distributed (approximately 20 times more living than dead), we believe that confusion matrix analysis is much more important compared to accuracy and ROC.

However, we will still provide a brief explanation with regards to accuracy and ROC.

*Accuracy*

Using J48 Decision Trees as the classifier model and CorrelationAttributeEvaluation or InfoGainAttributeEvaluation algorithms as attribute selection produced the best results in terms of accuracy, with 98.30% accuracy. Overall, J48 Decision Trees was the best classifier model across all five attribute selection algorithms, consistently producing the highest accuracy. The next best classifier model was Naive Bayes, and the worst classifier model was OneR, which is to be expected since OneR only takes into account one attribute. With regards to attribute selection algorithms, Correlation Attribute Evaluation, InfoGainAttributeEvaluation, and CfsSubset all worked very well with insignificant differences between the three. Surprisingly, our intuition worked better than GainRatioAttributeEvaluation.

In terms of overall accuracy, we had very high accuracy numbers, with everything above 96%. However, we must note that since we have a skewed distribution of class labels, high accuracy is less significant in this context.

*Confusion Matrix*

Before we determine which algorithms were the best based on confusion matrices, we will first discuss our metric for determining the best algorithm using the confusion matrix. We refer to the following table, which also appears on page 12.

|  | death\_yn = y (predicted) | death\_yn = n (predicted) |
| --- | --- | --- |
| death\_yn = y | a | b (false negative) |
| death\_yn = n | c (false positive) | d |

We determined the best algorithm based on the least number of false negatives (value b in the table). This is because false negatives are the most fatal result, as in someone who will die from COVID is predicted to not die and thus will not take any precautions in preventing death. Using this metric, the best model was using NaiveBayes as the classification algorithm and CfsSubset for attribute selection. Overall, the NaiveBayes classification algorithm performed the best, while the OneR classification algorithm by far performed the worst, with almost 50% predicted false negatives of death cases. There were no significant differences produced by different attribute selection algorithms, with the exception of GainRatio and our intuition performing significantly worse. It is to be expected that our intuition performs worse.

*Receiver operating characteristic (ROC) Curve*

The model with the best ROC curve results used DecisionTables as the classification algorithm and InfoGain as the attribute selection algorithm. The area under the ROC curve is 0.9898. In terms of attribute selection, CorrelationEval, InfoGain, and CfsSubset performed significantly better than GainRatio and our intuition. In terms of classifier algorithms, OneR performed significantly worse than the other three algorithms, which is expected since OneR only takes into account one attribute. The ROC curve is a visualization of model’s accuracy as the discrimination threshold is varied; the larger the area under the curve, the better the model is at classifying when an inherent bias towards one result or another is included.

***Best Model***

The model with the best accuracy used **J48** as the classifier algorithm and **CorrelationAttributeEvaluation** or **InfoGainAttributeEvaluation** algorithms as attribute selection. It produced a model with 98.30% accuracy.

The model with the best true positive rate (i.e. the model with the highest percentage of correctly identified positives) used **NaiveBayes** as the classification algorithm and **CfsSubset** as the attribute selection algorithm. This model correctly identified 2491 as lethal while misidentifying 442 fatal COVID cases as nonfatal (A false negative). Thus, the model has a 84.93% true positive rate.

The model with the best false positive rate (i.e. the model with the lowest percentage of misclassified negatives) used **J48** as the classification algorithm and **GainRatio** as the attribute selection algorithm. This model correctly identified 54936 COVID cases as nonlethal while accidentally flagging 62 nonlethal COVID cases as lethal (a false positive). Thus, this model has a 0.1127% false positive rate.

The model with the best ROC curve results used **DecisionTables** as the classification algorithm and **InfoGain** as the attribute selection algorithm. The ROC value is 0.9898.

In real life, a false positive corresponds to a patient being falsely flagged as high risk and receiving more intensive care than necessary. A false negative corresponds to a patient being falsely flagged as low risk and receiving less intensive care than necessary. Thus, a false negative is much more costly than a false positive. Therefore, the false negative rate, accuracy, and area under the ROC curve are less important than the true positive rate. We conclude that the model that used **NaiveBayes** as the classification algorithm and **CfsSubset** as the attribute selection algorithm is the best for general use. This model produced 442 false negatives, which accounts for about 13% of total death cases.

**Conclusion and How to Reproduce Our Model**

***Conclusion***

Our project goal was to predict if patients would die from COVID from their personal data, such as county and state residence, age, etc. Our dataset is from the CDC.

We first pre-processed our data with python, using the Pandas DataFrame. Our preprocessing included removing any instances that were missing class labels, removing duplicate values, and removing derivable attributes. Then, we converted the .csv file (original data format) to a .arff file (WEKA data format) to prepare for processing in WEKA.

We trained 20 models, with 5 attribute selection algorithms and 4 classification algorithms, in WEKA. Our attribute selection algorithms were Correlation Attribute Evaluator, Information Gain Attribute Evaluator, Cfs Subset Evaluator, Gain Ratio Attribute Evaluator, and our own intuition. Our classification algorithms were NaiveBayes, J48 (decision tree), OneR, and DecisionTable. We also split data into testing and training in a 33:67 ratio. We used stratified sampling to guarantee a representative sample.

We analyzed our results using three metrics: accuracy, confusion matrix (includes TP, FP), and area under ROC curves. In terms of accuracy, all models produced results in the range 96% to 99%. In terms of ROC curves, values ranged from 0.6 to 0.99. In terms of the confusion matrix, we evaluated the best model through the least number of false negatives, since this would most negatively affect the patient. Again, we saw similar results, with the exception of using the GainRatio and our intuition as attribute selection algorithms (these algorithms produced significantly worse results). In this dataset, the ratio of non-deaths to deaths is approximately 20:1. Thus, the confusion matrix metric is a much better evaluator of models compared to accuracy. Using the confusion matrix metric, our best model is using NaiveBayes as the classification algorithm and CfsSubset as the attribute selection algorithm. This model produced 13% of false negatives within all the death cases.

Our project is significant because it is relevant in current times with the ongoing COVID pandemic. Our model can be used to predict whether or not patients will die from COVID. The practical application is that if a patient is predicted to die from COVID, they would be aware of this and can take extra precautions to prevent fatality.

***Project Takeaways***

Through this project, we explored various data selection and model creation algorithms in depth. In particular, by examining machine learning through the lens of a real-world problem, we grew a deeper understanding of the nuance involved in algorithm and model selection. By examining the impact of a false negative versus a false positive, we learned about the different lenses through which a model can be judged; in addition to accuracy, true positive rates, false negative rates, and area under curve values can be used to ascertain whether a model will be useful in the world. For example, in the context of COVID lethality rates, the true positive rate is more significant than accuracy, though the false negative rate is less important, since the cost of a false negative is relatively low.

On the other, more technical hand, we learned to implement various data preprocessing steps in both WEKA and Python. We also learned to use many attribute selection algorithms and to create various models using different algorithms.

***Model Reproduction***

In WEKA, open training.arff in the CfsSubset folder. Under the Classify tab, click NaiveBayes (bayes -> NaiveBayes) to set it as the classifier. Under test options, click Supplied test set and set testing.arff as the test set. Then, click start. This will reproduce our best model.

**Team Members and Tasks Performed**

Isabella Zhu,

Developed Python code to remove instances with missing class variables

Checked for redundancy (duplicated instances) in data during code preprocessing

Selected attributes using CFS Subset Evaluation and Gain Ratio Attribute Evaluation

Analyzed accuracy tables and confusion matrices to identify the best model

Wrote the presentation

Lilian Zhu,

Removed derivable attributes using WEKA software

Applied WEKA’s ReplaceMissingValues during data preprocessing

Selected attributes using Correlation Attribute Evaluation and Information Gain Attribute Evaluation

Analyzed accuracy tables and confusion matrices to identify the best model based on true positive and false negative rates

Wrote this section

Isabella Zhu and Lilian Zhu,

Chose a real-world dataset

Defined our data mining goal

Evaluated attributes for derivable attributes

Preprocessed data for use in predictive modeling

Selected attributes using intuition and past knowledge of epidemiology

Split data sets into test and training sets

Verified the quality and class distribution of the test/training split to ensure that both are representative of the original dataset

Implemented model classifiers (Naive Bayes, J48, OneR, and DecisionTable) in WEKA

Added an appendix with screenshots verifying the results

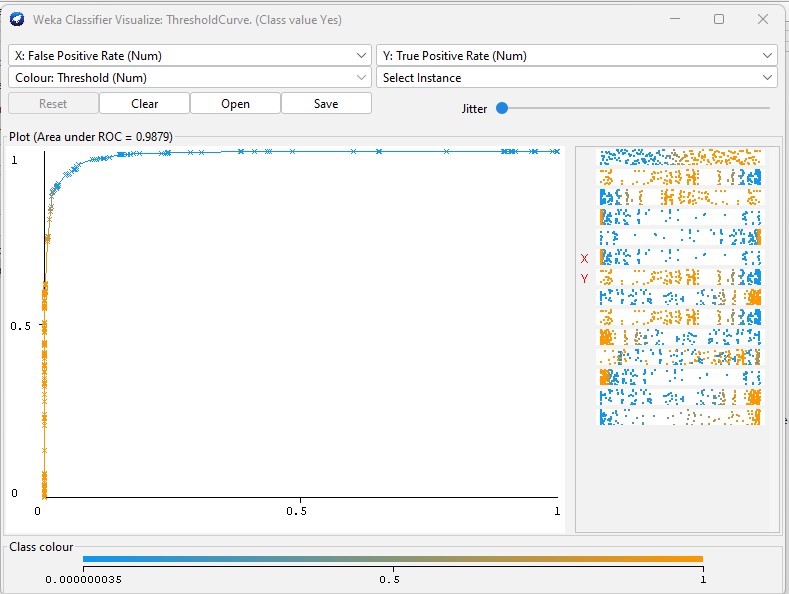
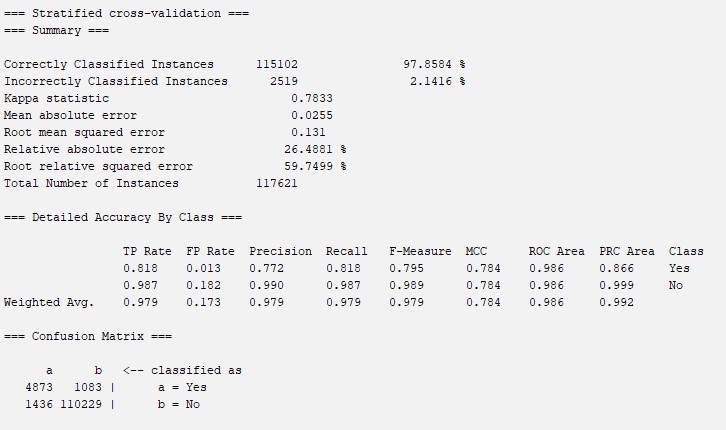
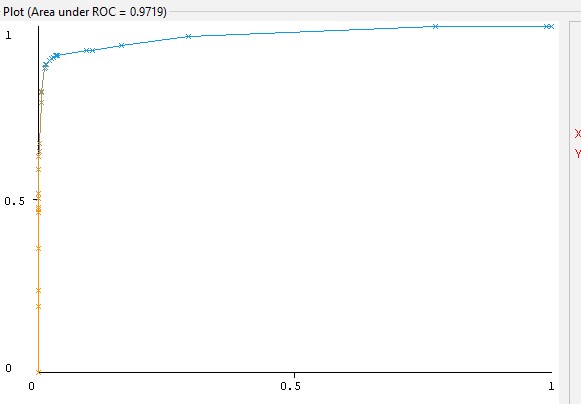
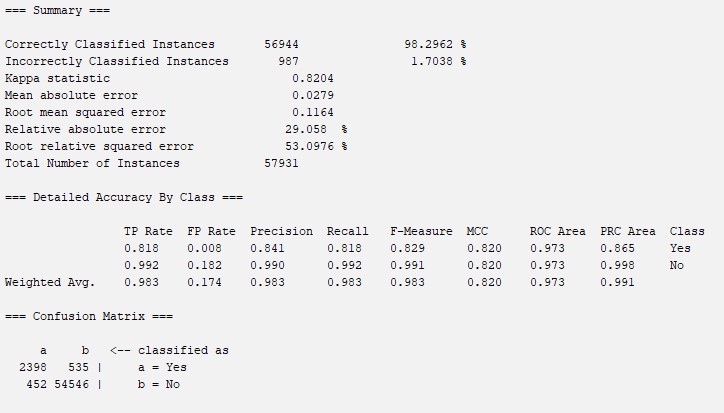
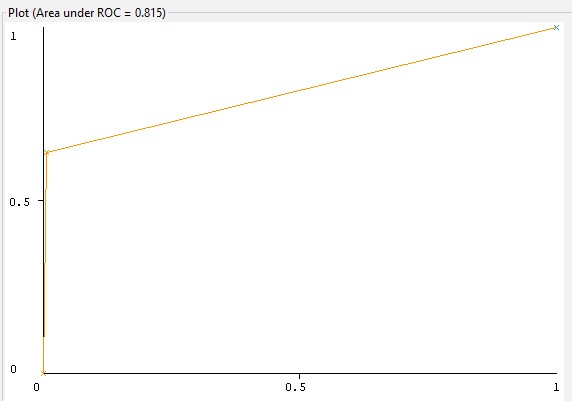
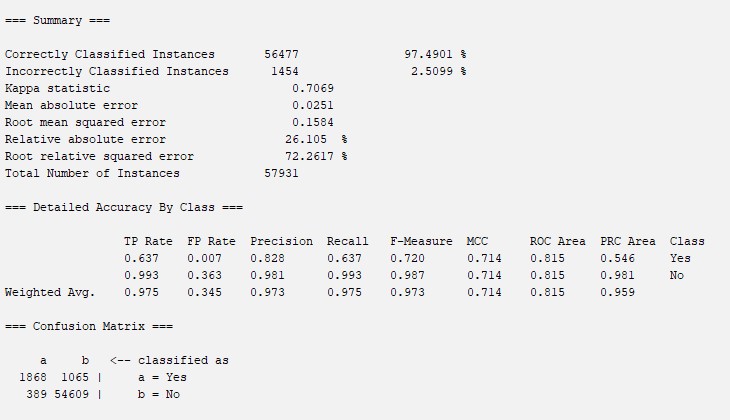
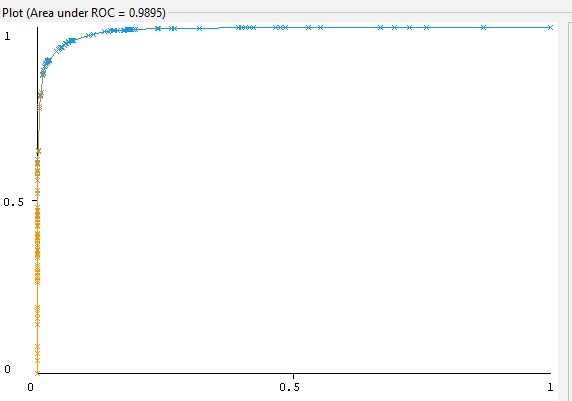
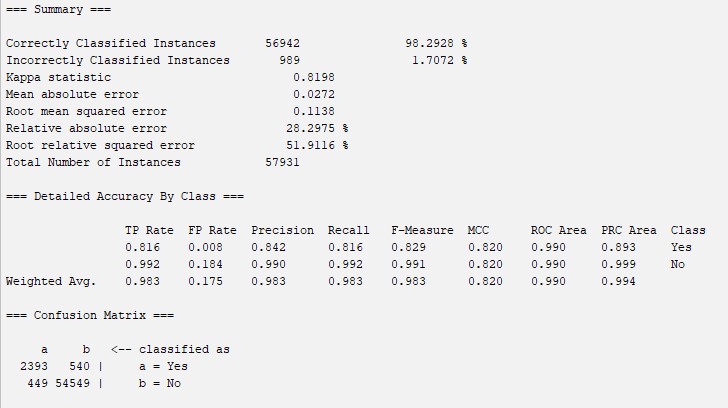
**Sources**

CDC Data, Analytics and Visualization Task Force, & Lee, B., COVID-19 Case Surveillance Public Use Data with Geography (2022). Centers for Disease Control and Prevention. Retrieved October 7, 2022, from <https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data-with-Ge/n8mc-b4w4>.

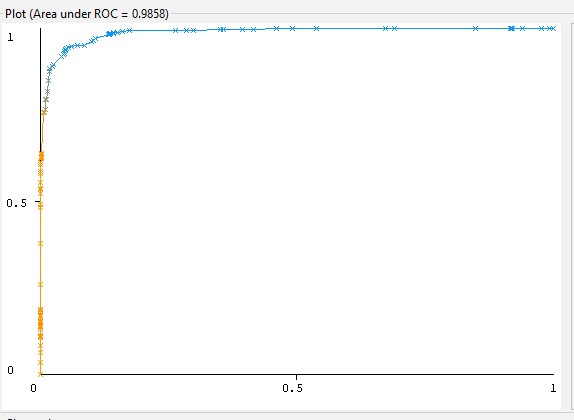
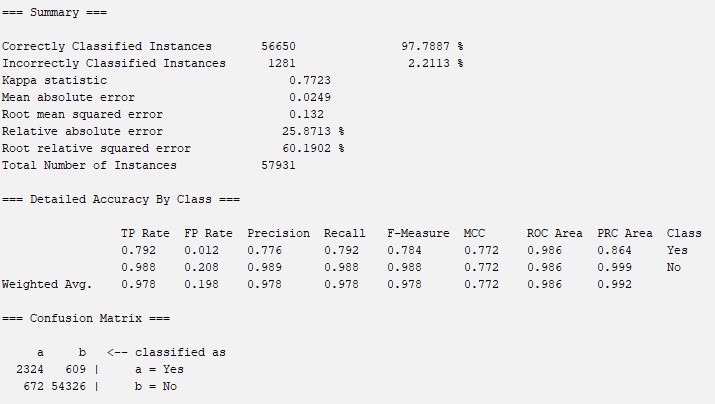
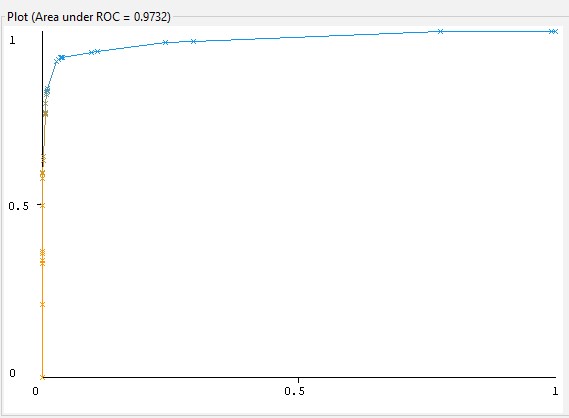
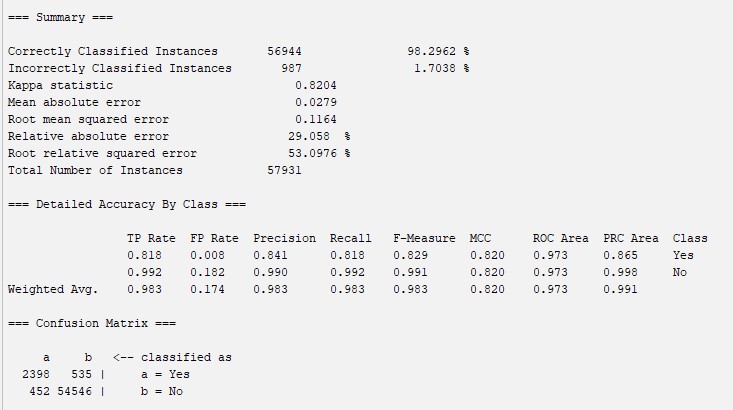
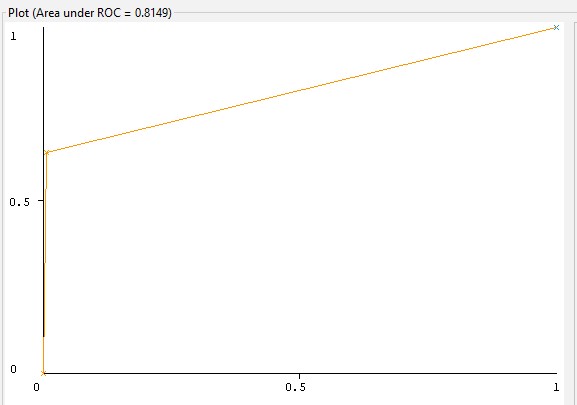
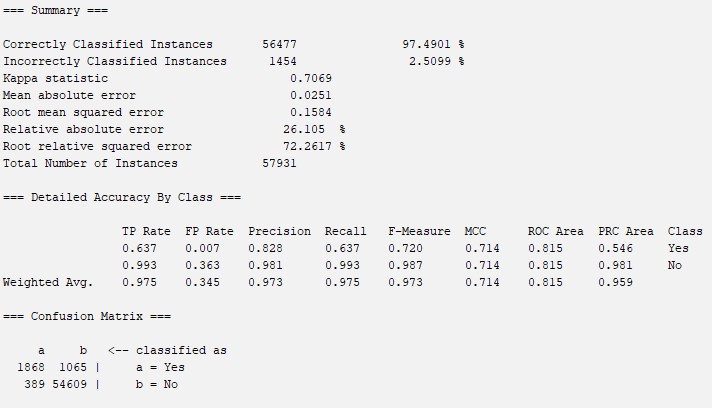
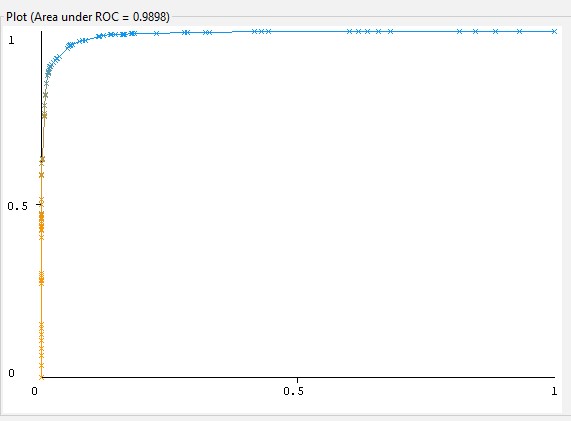
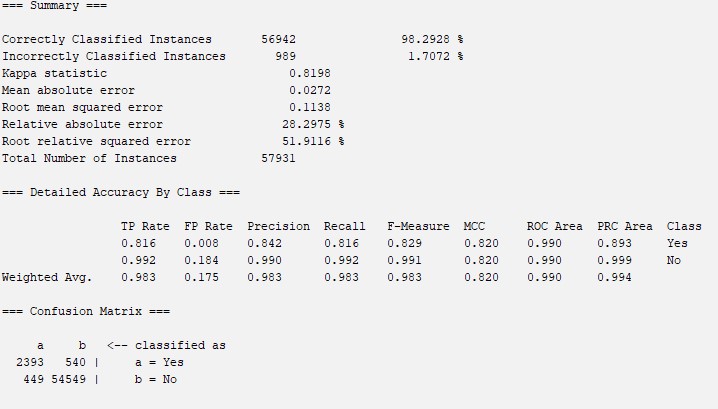
**Appendix**

These are the screenshots of our models’ results. The top screenshot includes the confusion matrix; the bottom shows the Area Under Curve

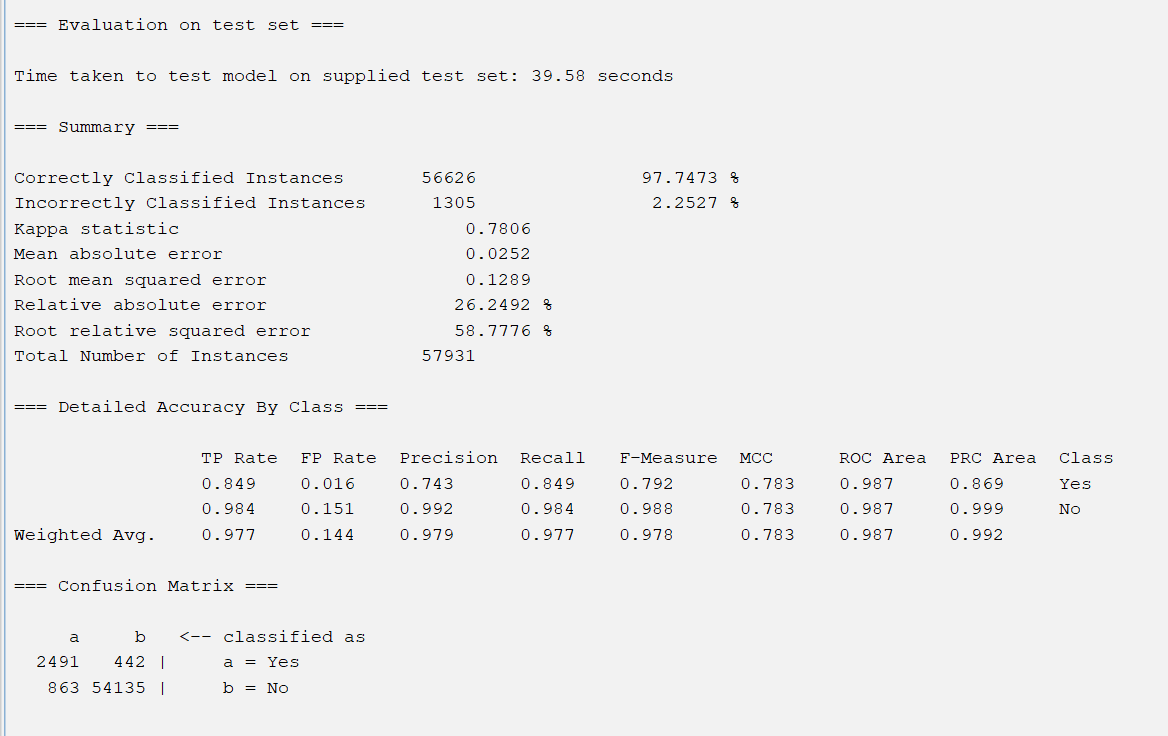
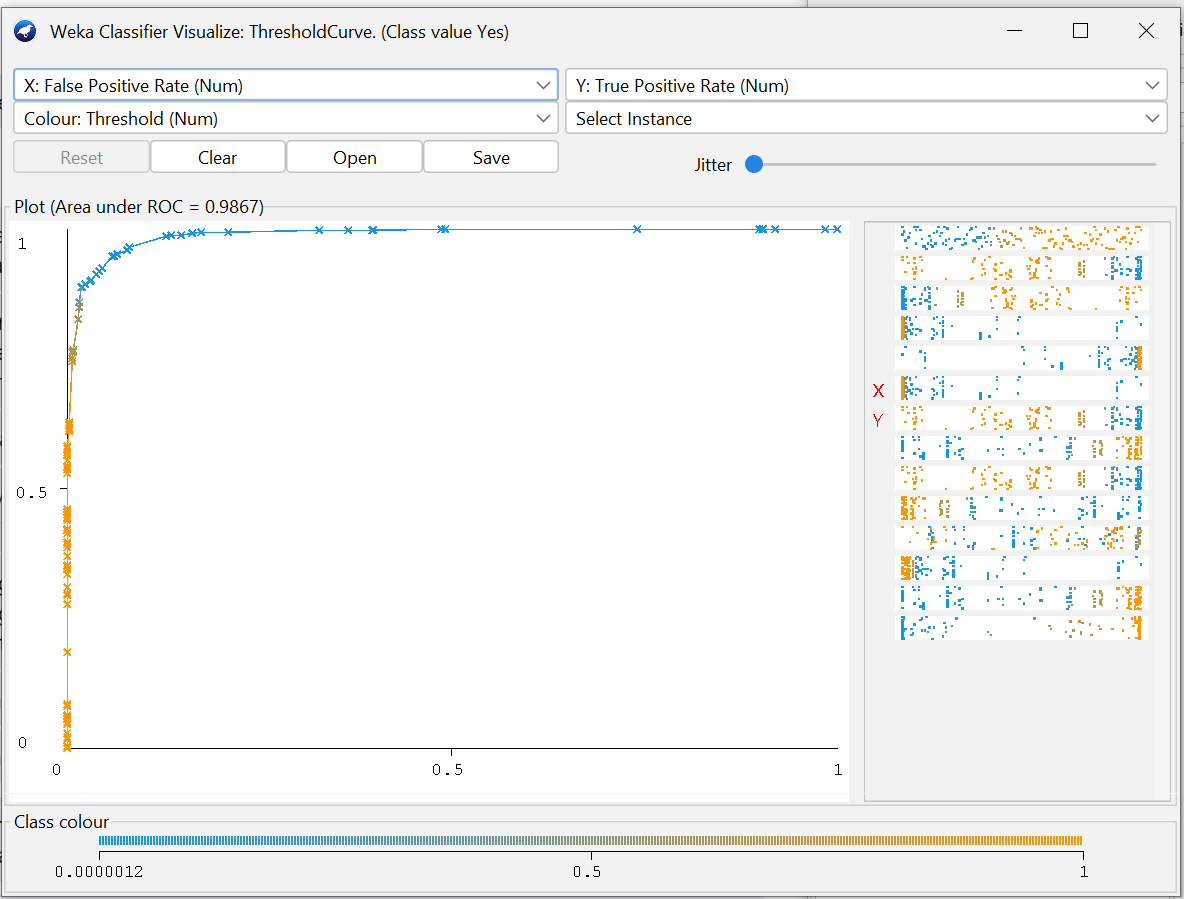
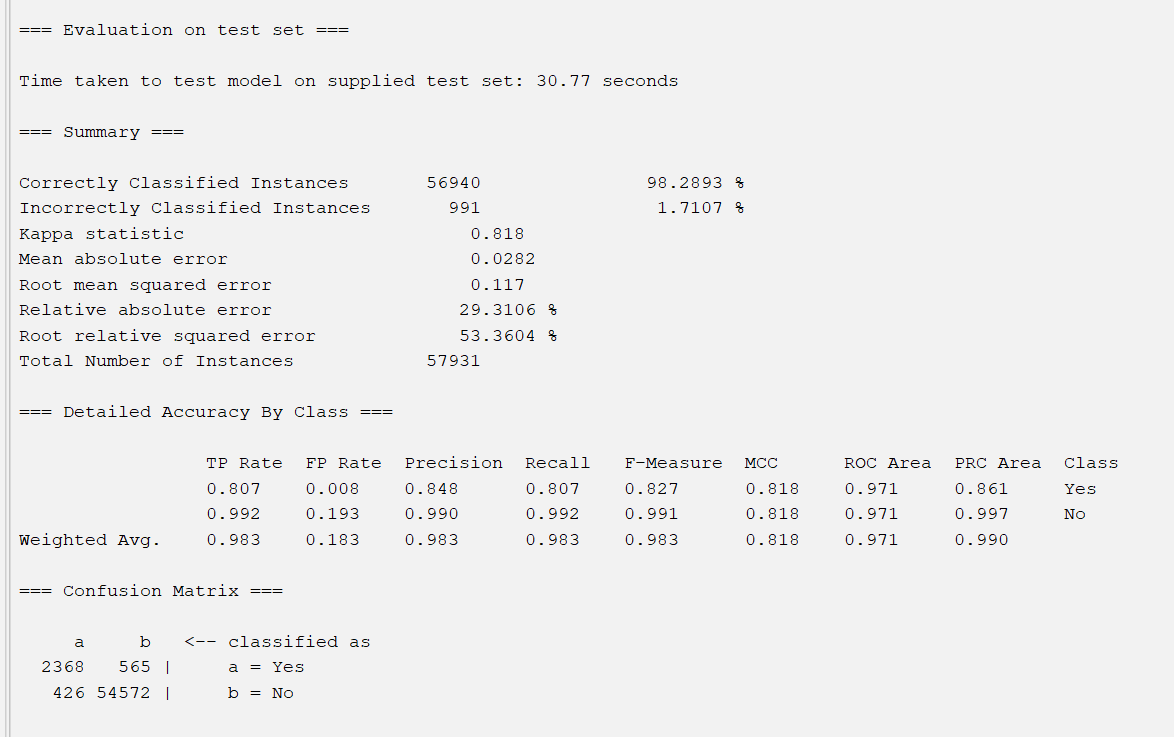
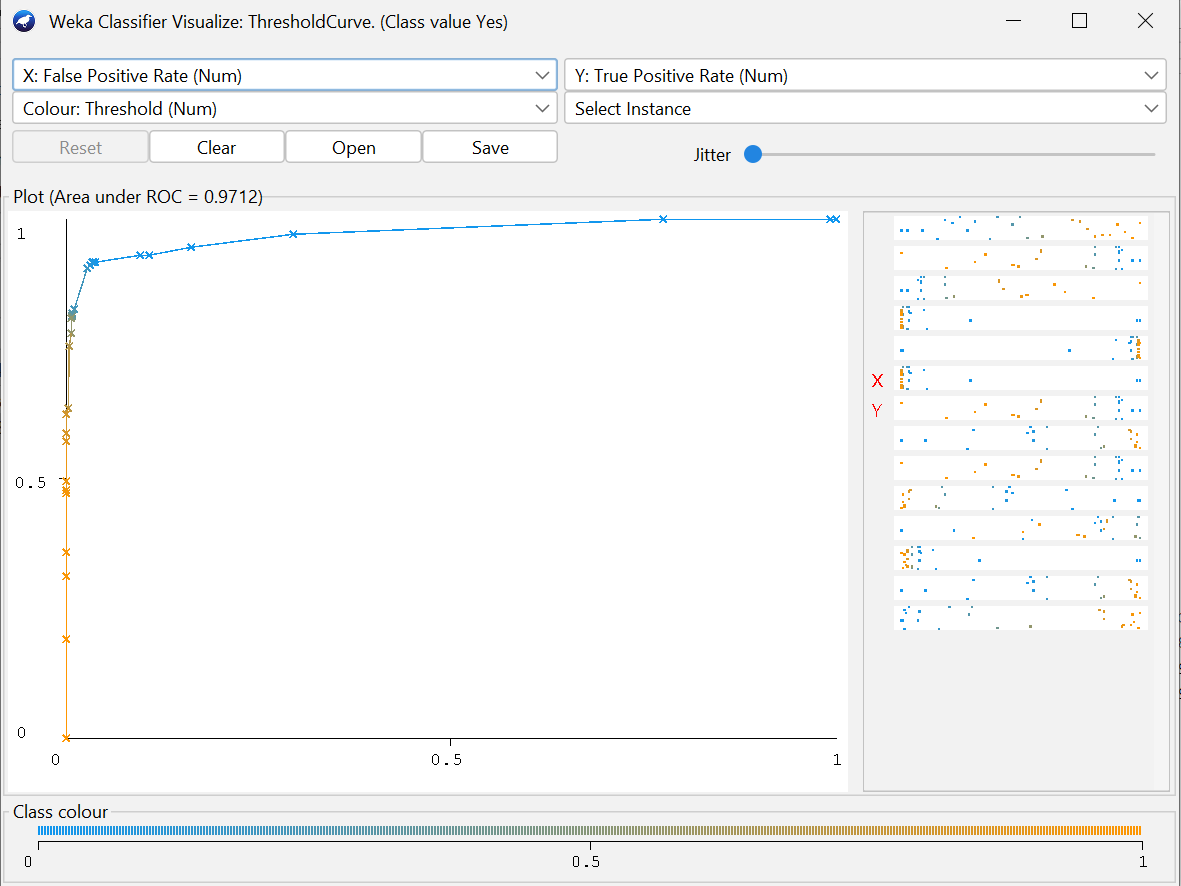
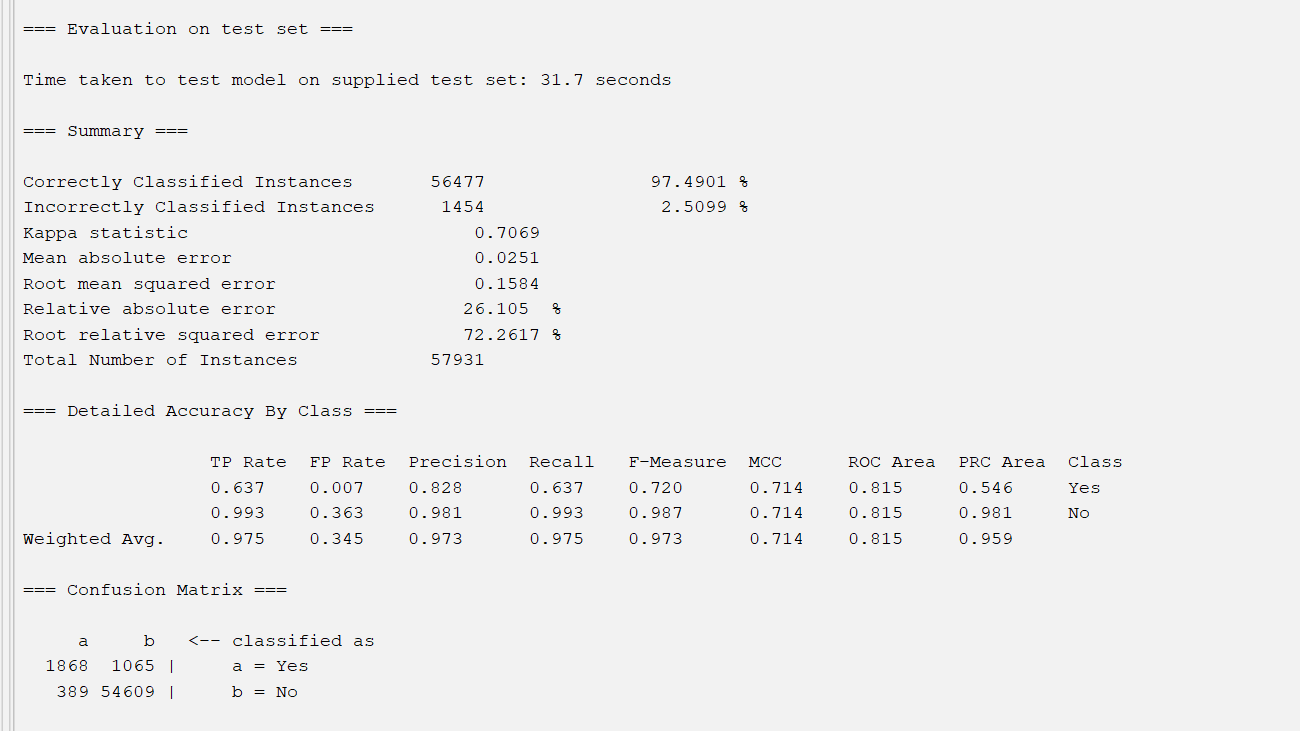
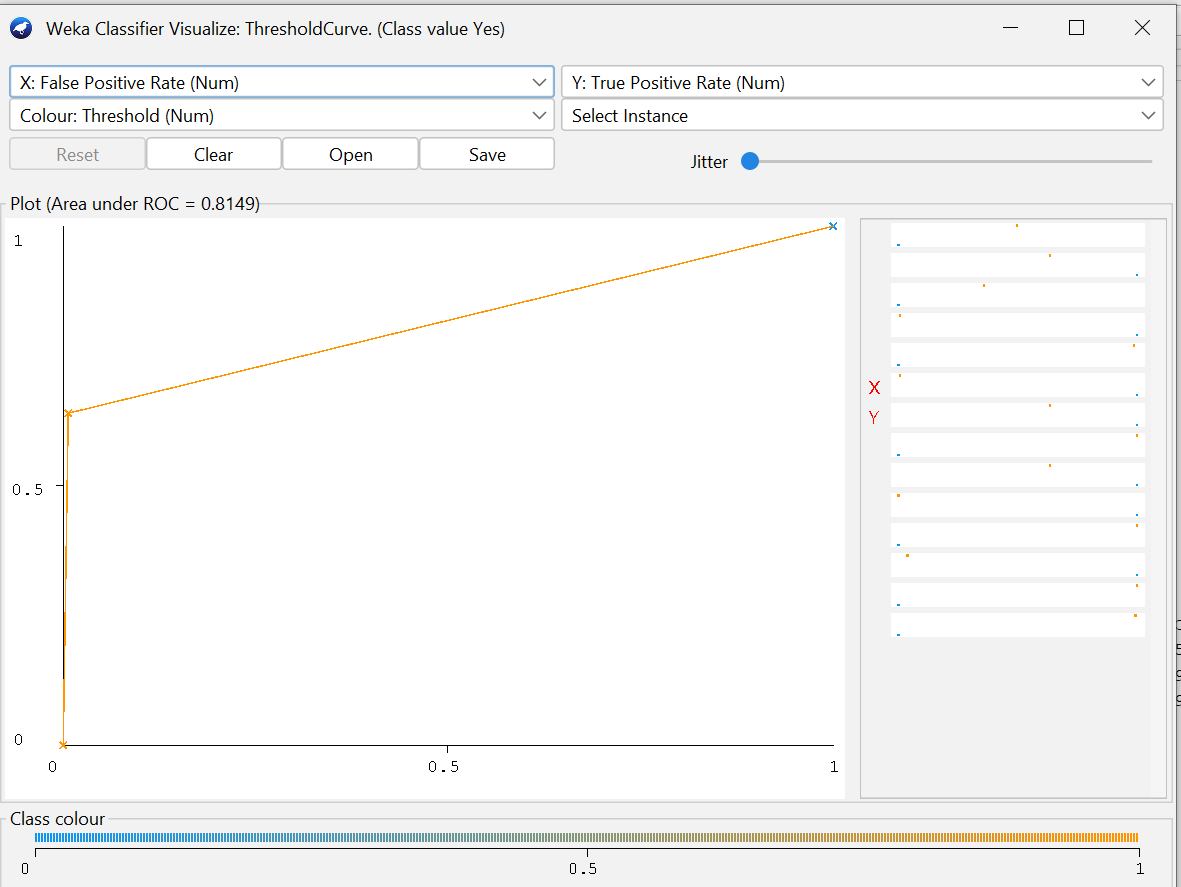
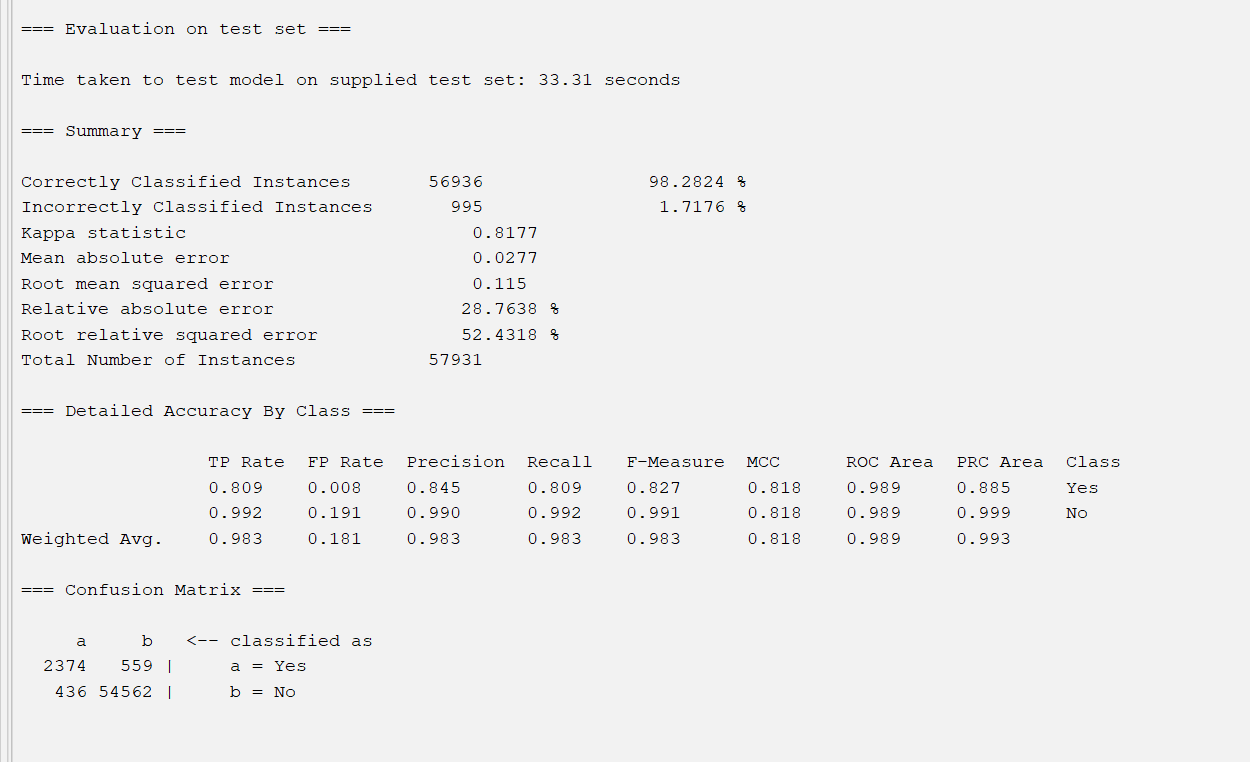
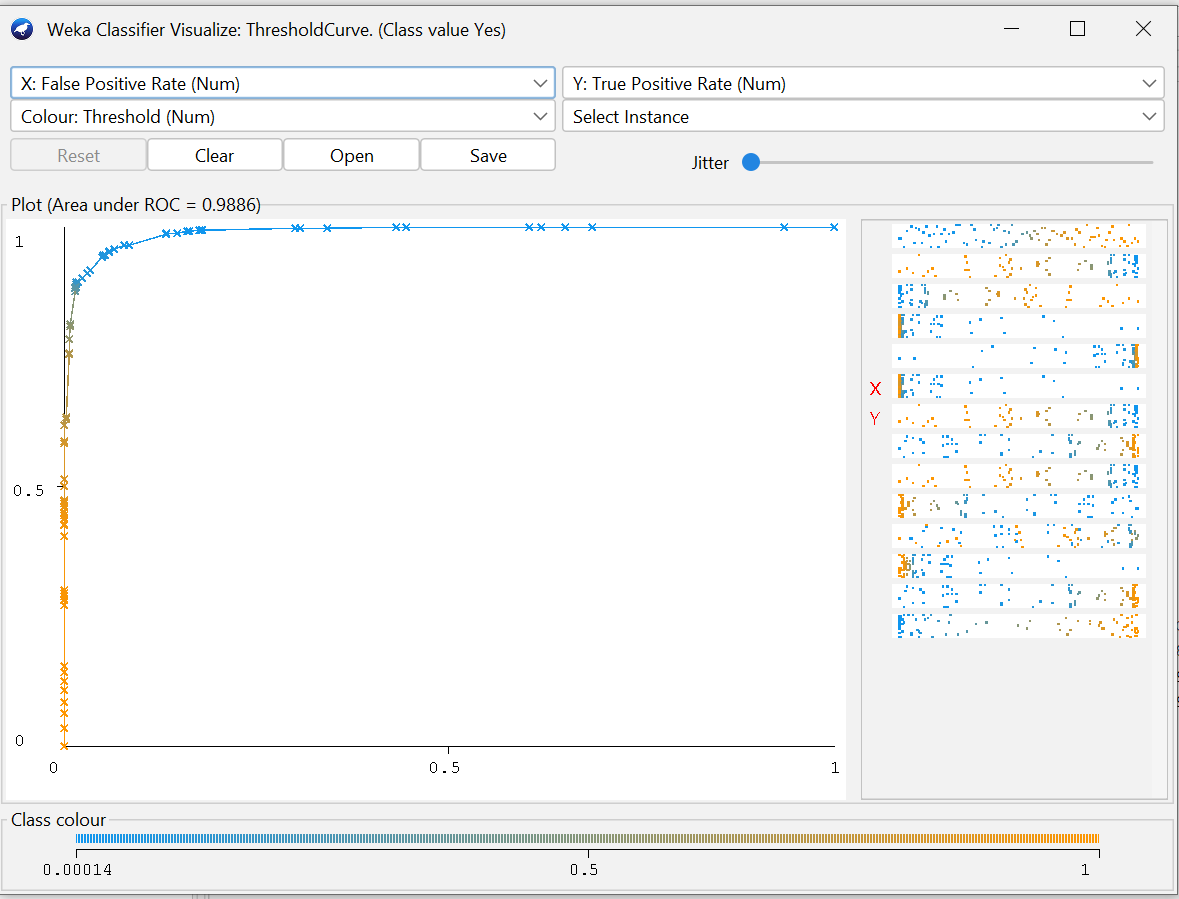
CorrelationAttributeEval Outputs

* Naive Bayes
* J48
* OneR
* Decision Table

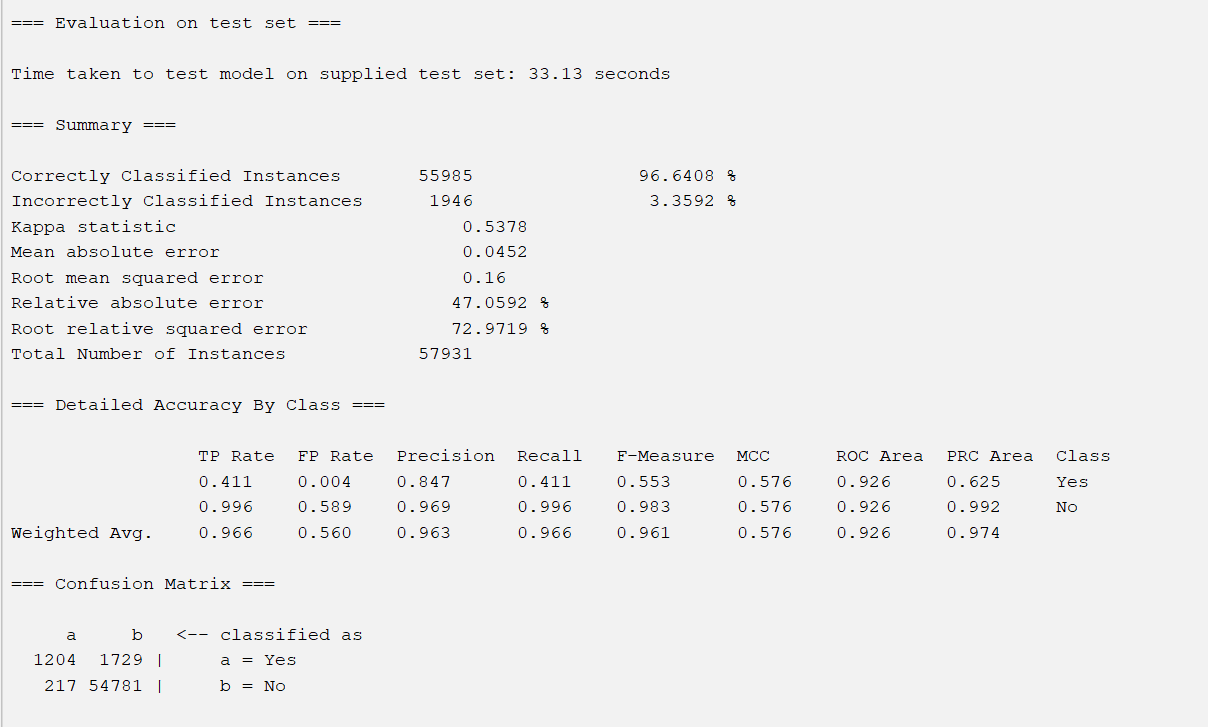
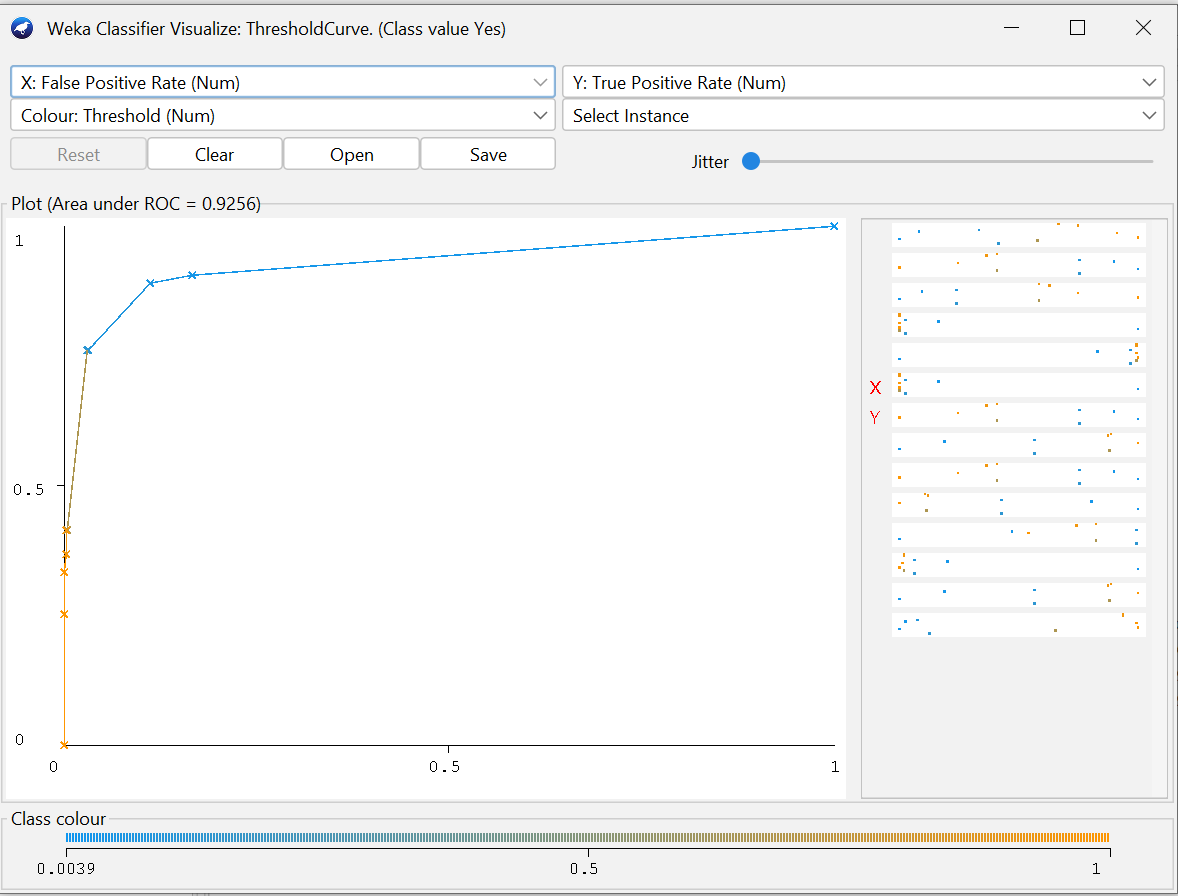
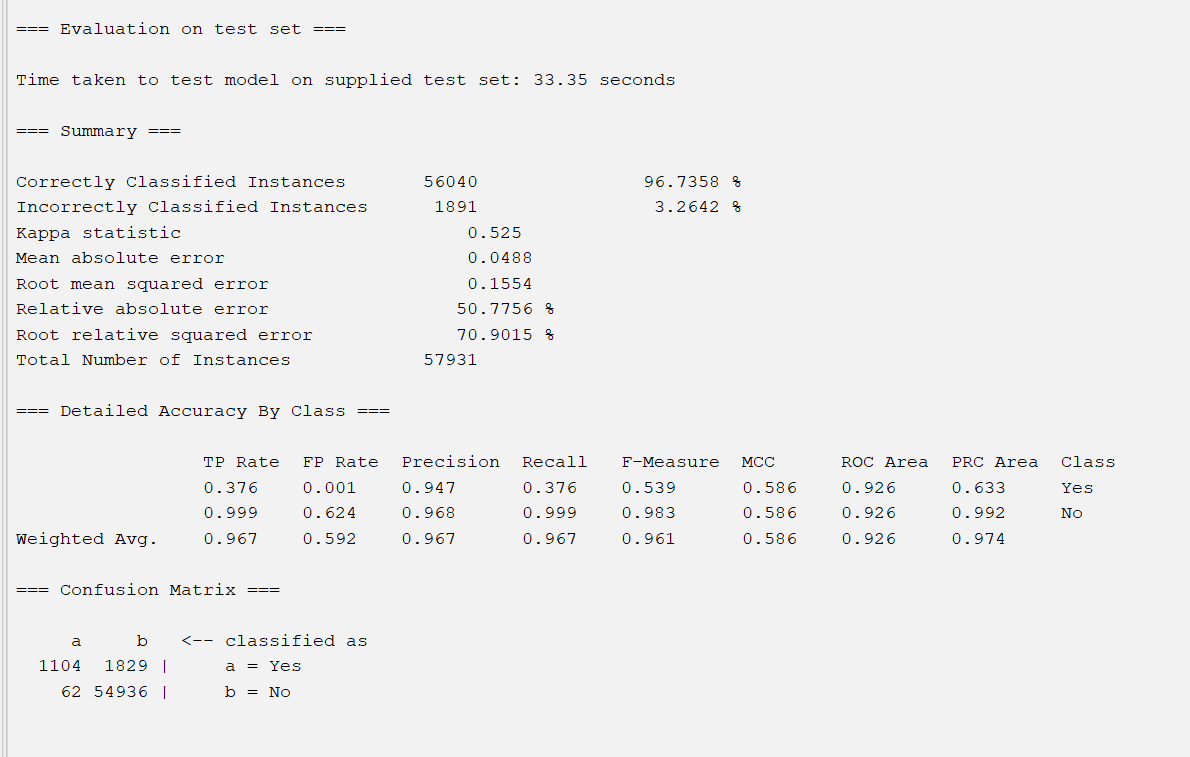
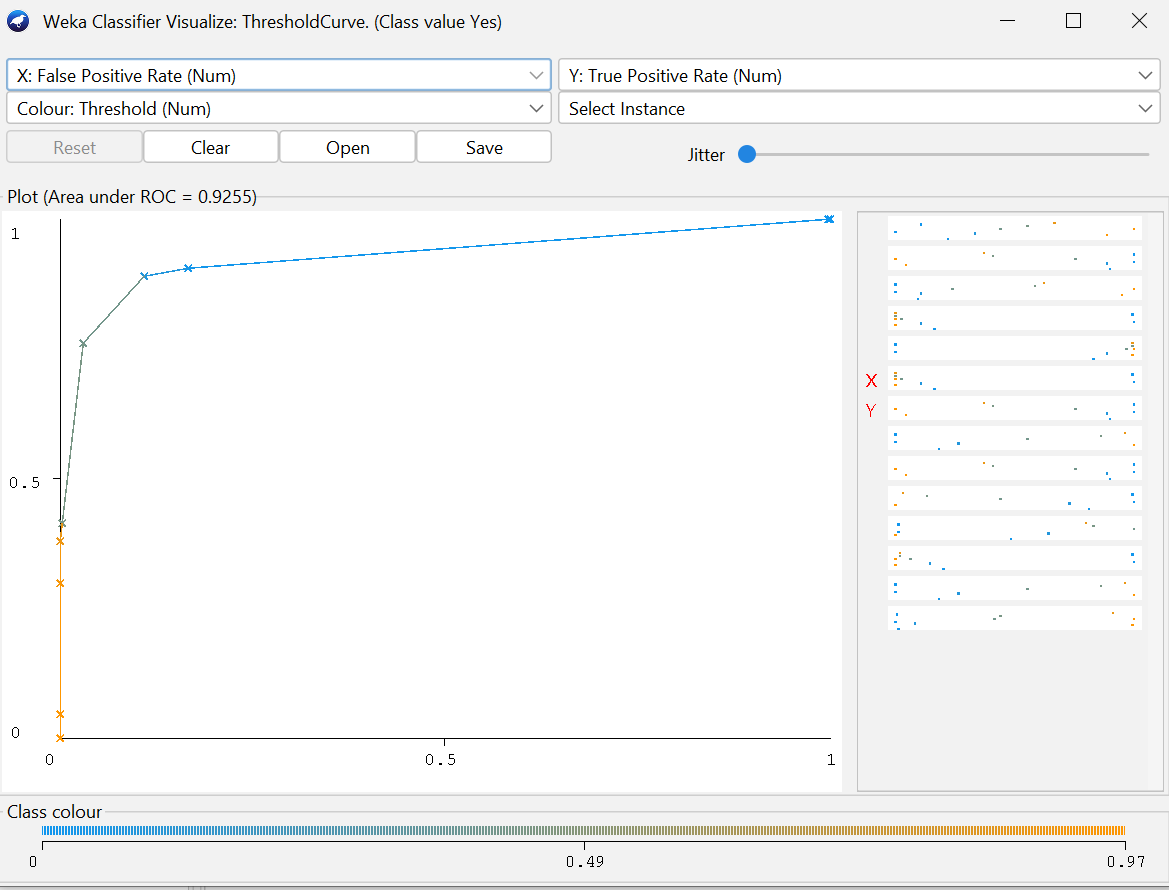
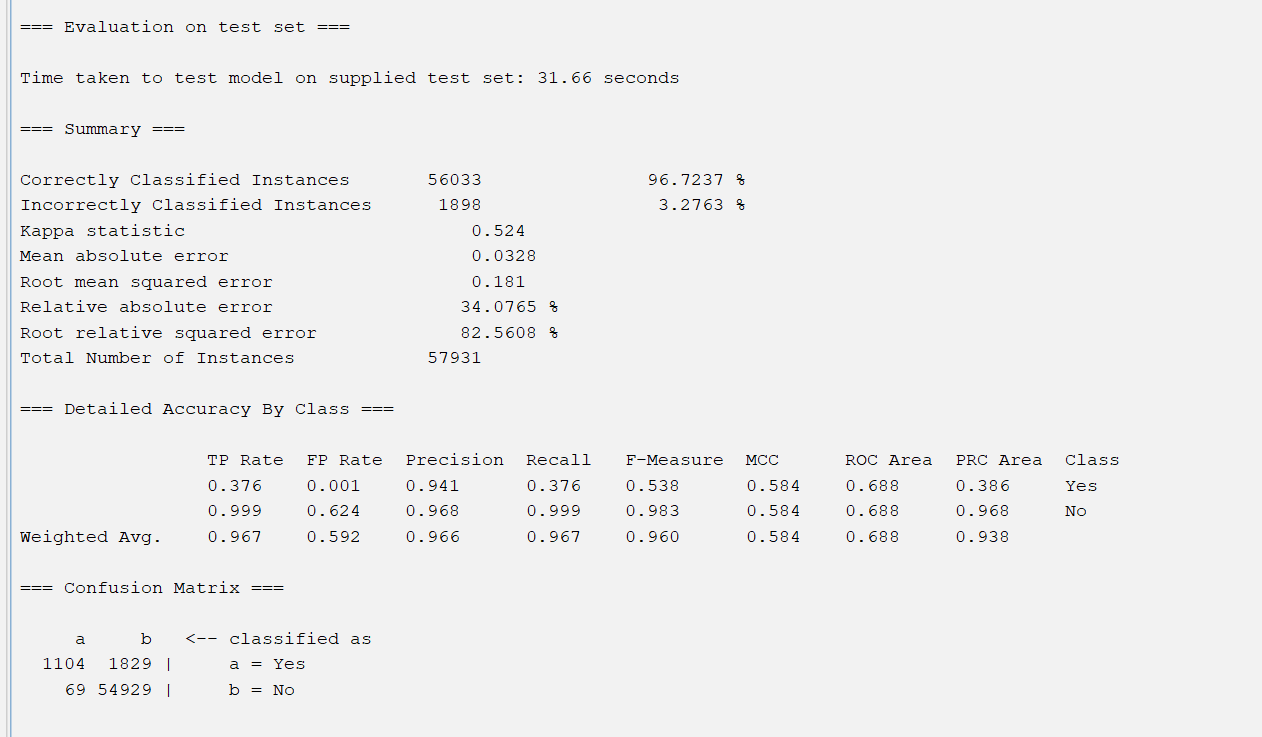
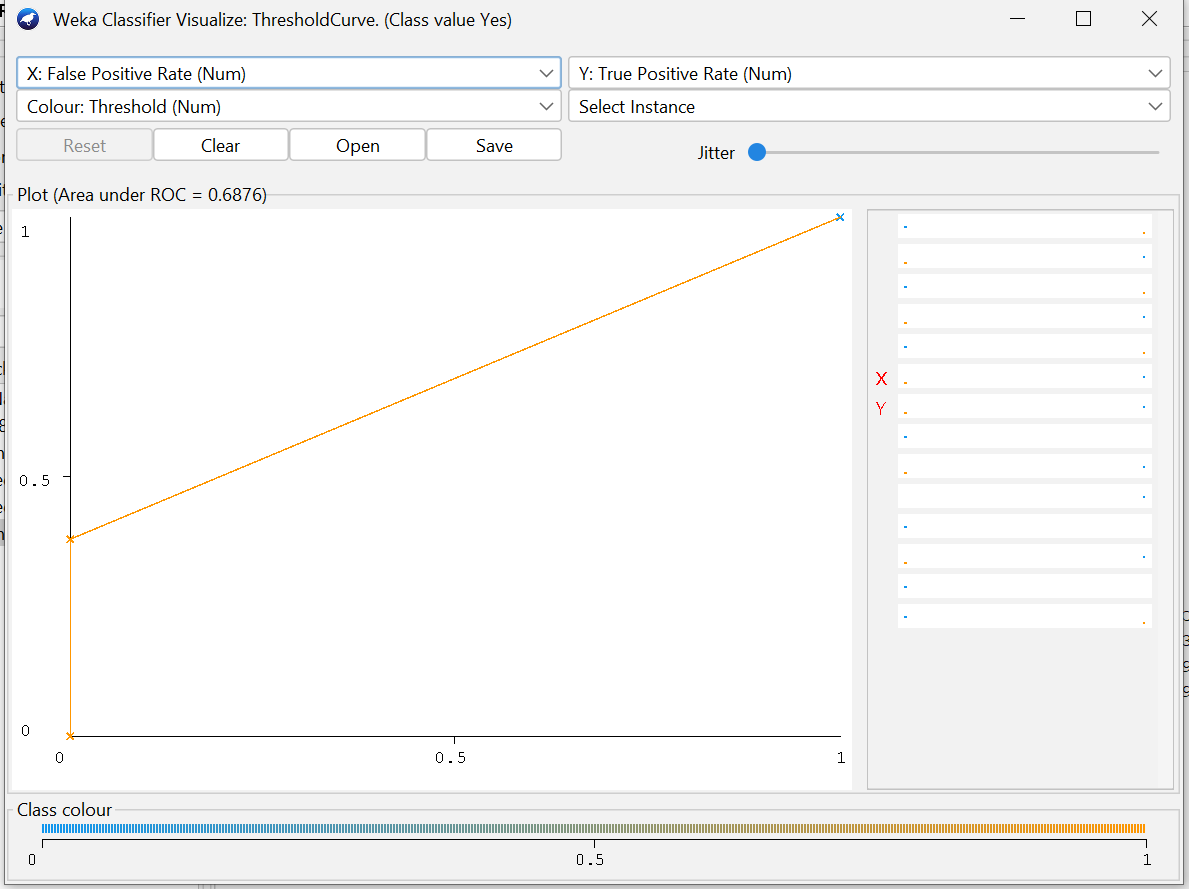
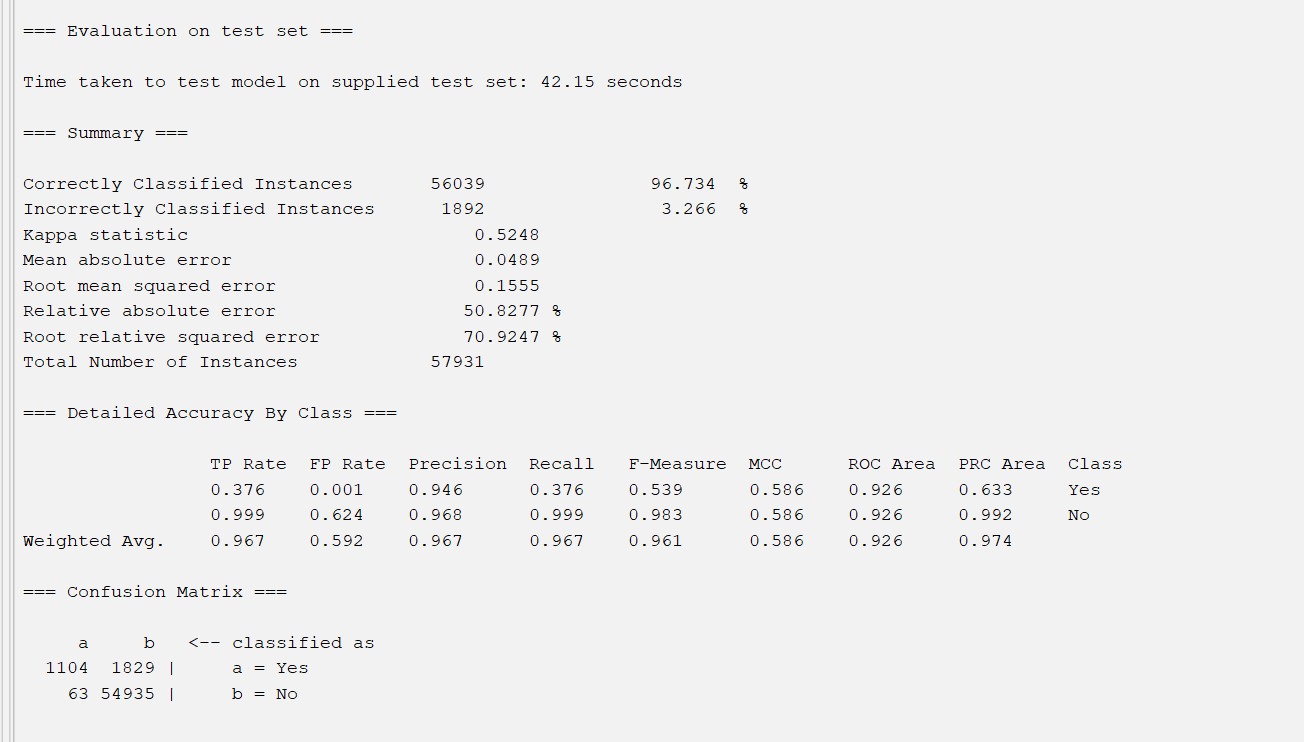
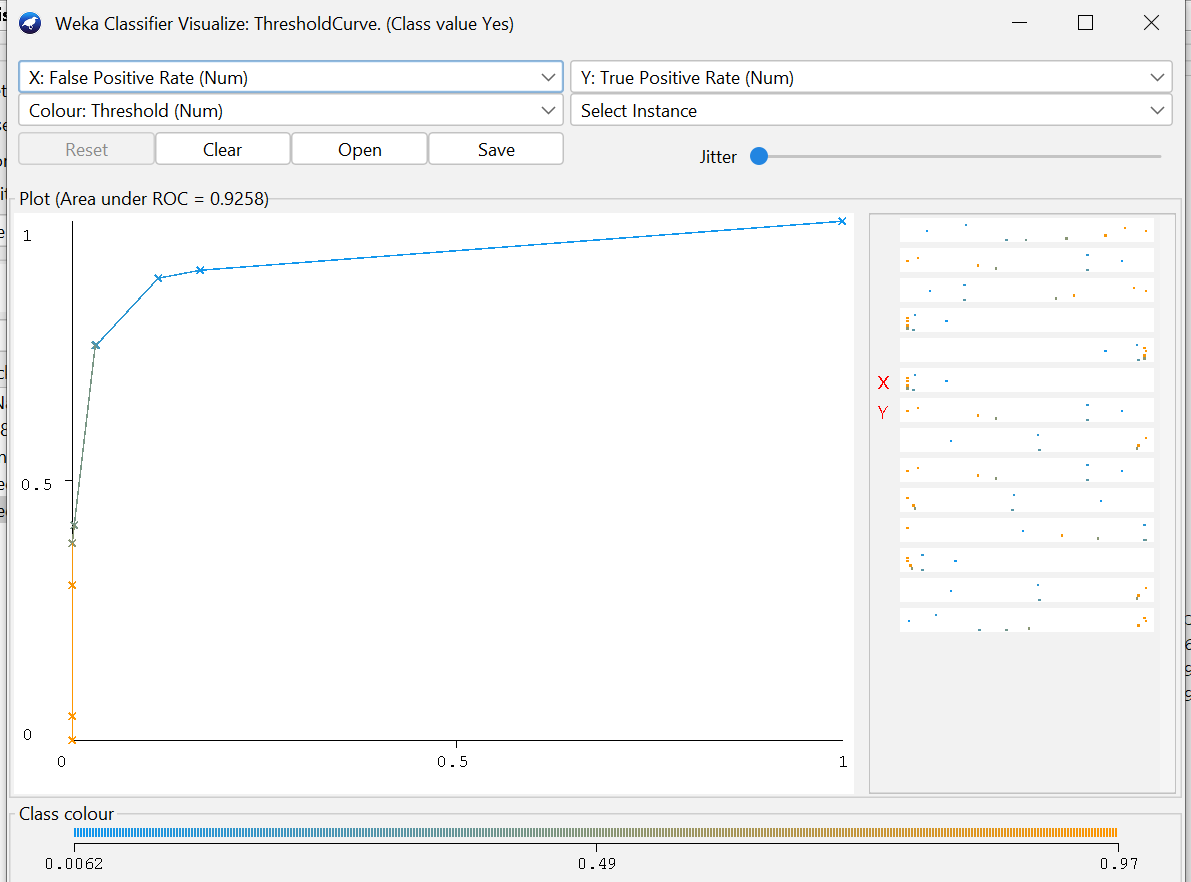
InfoGainAttributeEval Outputs

* Naive Bayes
* J48
* OneR
* Decision Table

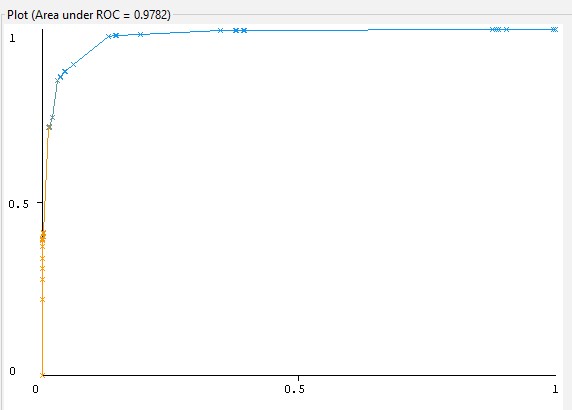
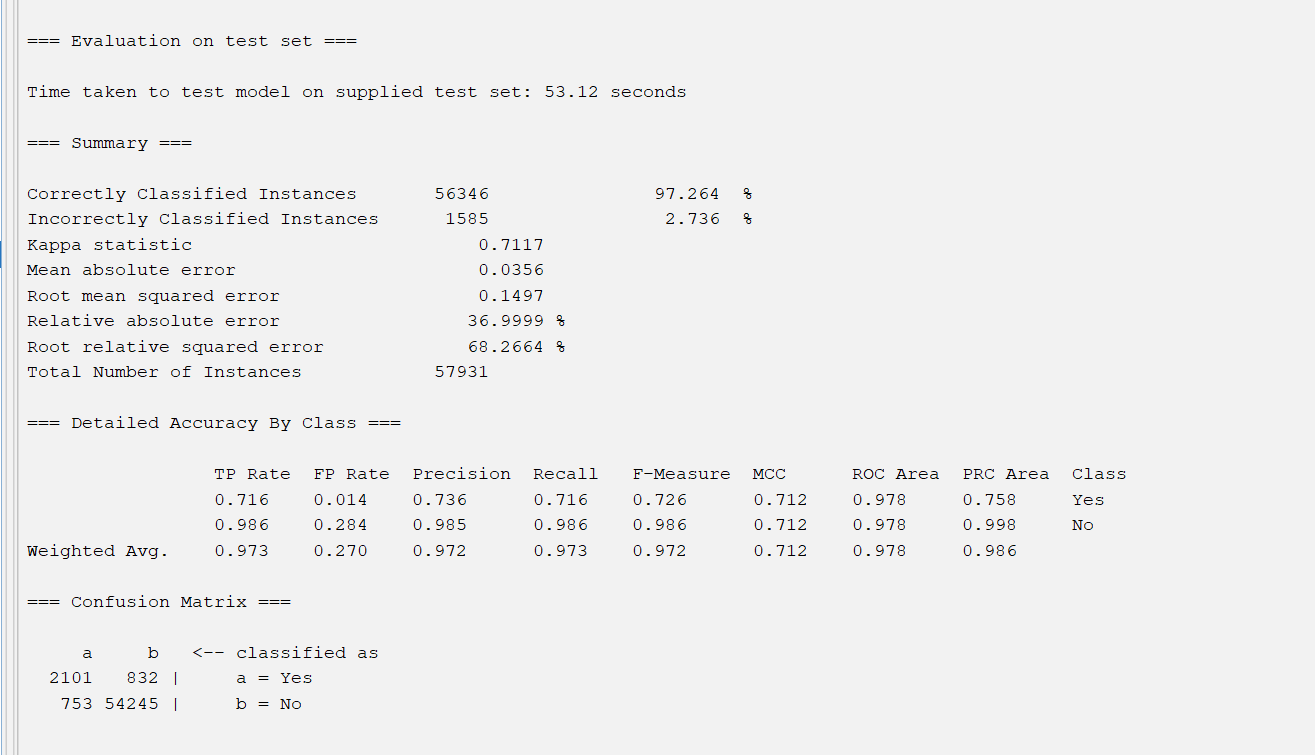
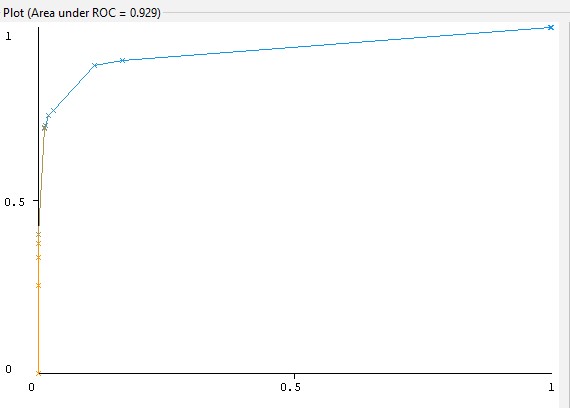
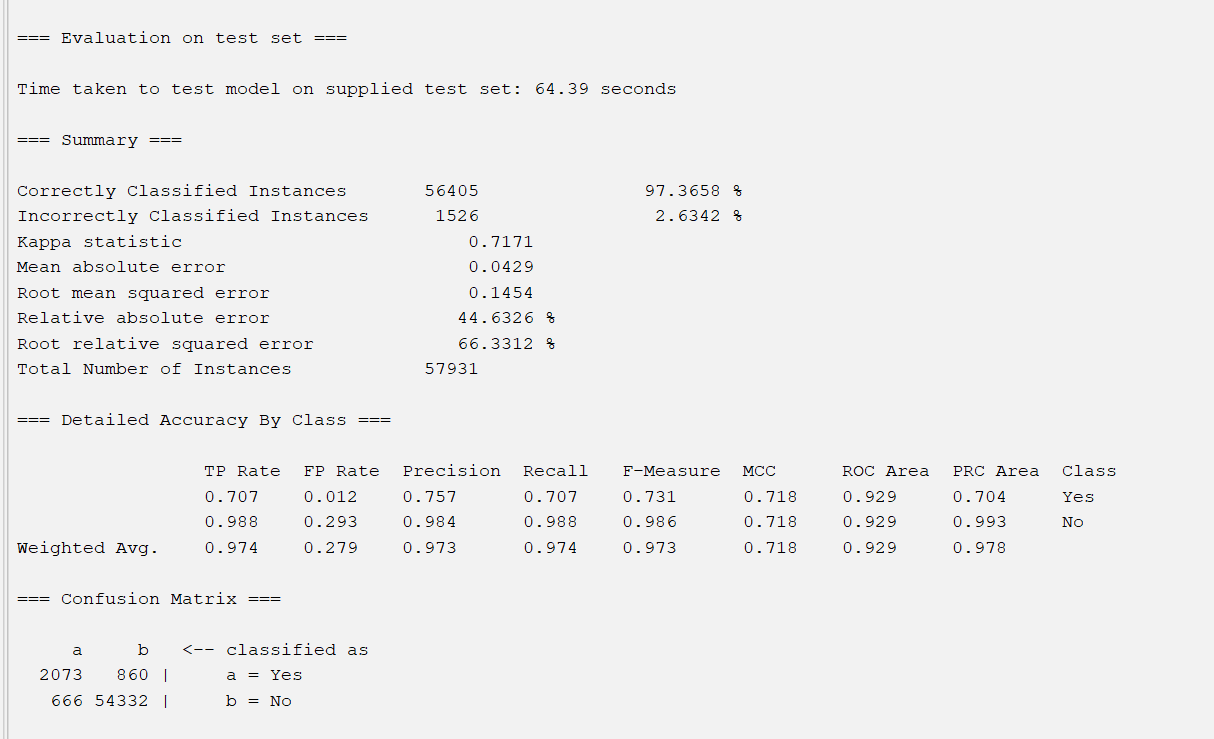
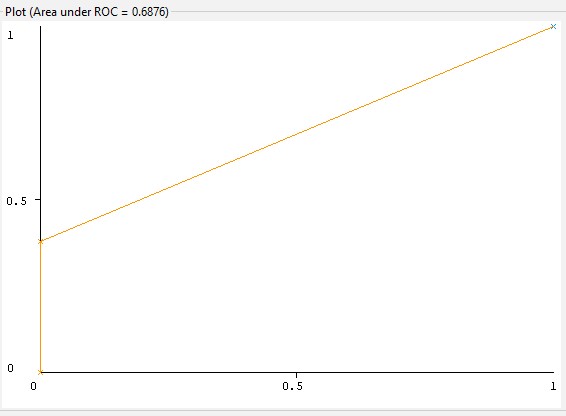
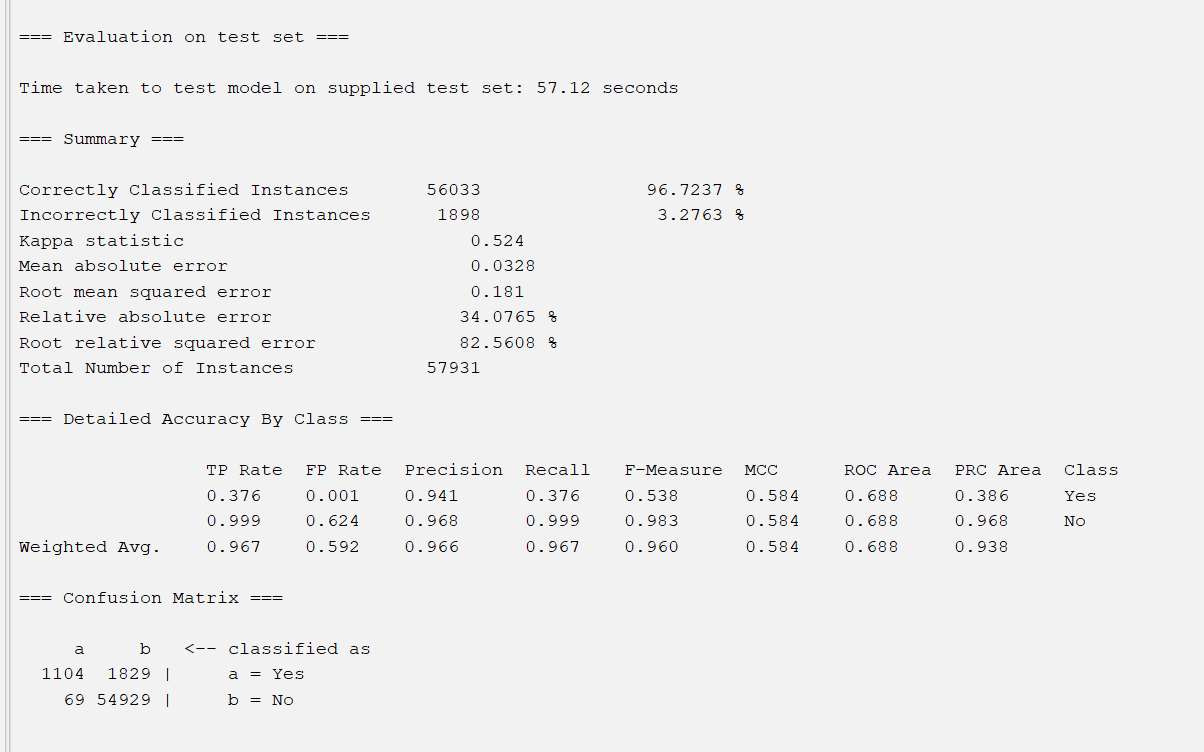
CfsSubsetEval Outputs

* Naive Bayes
  + 
  + 
* J48
  + 
  + 
* OneR
  + 
  + 
* Decision Table
  + 
  + 

GainRatioAttributeEval Outputs

* Naive Bayes
  + 
  + 
* J48
  + 
  + 
* OneR
  + 
  + 
* Decision Table
  + 
  + 

Intuition Outputs

* Naive Bayes
  + 
* J48
  + 
* OneR
  + 
* Decision Table
  + 