**Project Progress**

**CS 699 A1**

**Huijun Hong**

1. **Data Mining Goal**

Predicting whether tomorrow is a rainy day.

Weather prediction is an important task because it can be used to protect life and property. Forecasts based on temperature and precipitation are important to agriculture, and therefore to traders within commodity markets. It can also provide a better plan for people's travel plans.

1. **Detailed Description of the Dataset**

**Summary of the Dataset**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Brief Description** | **Number of Tuples** | **Number of Attributes** |
| <http://www.bom.gov.au/climate/data/> | Daily weather observations during 2007 to 2018 from different locations across Australia. | 145460 | 23 (including the class attribute) |

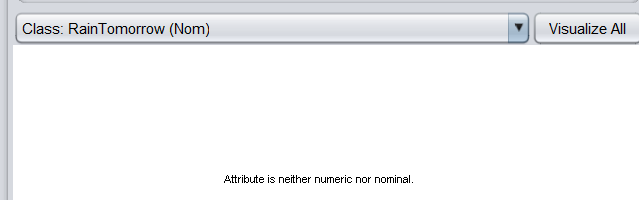
**Description of the Attributes**

|  |  |
| --- | --- |
| **Attribute Name** | **Brief Description** |
| **Date** | The date that the tuple belongs to. |
| **Location** | Different areas of Australia. |
| **minTemp** | Minimum temperature. |
| **maxTemp** | Maximum temperature. |
| **Rainfall** | The amount of rainfall recorded for the day (mm). |
| **Evaporation** | The amount of water evaporation. |
| **Sunshine** | The degree of sunshine. |
| **WindGustDir** | The direction of the strongest wind gust in the 24 hours to midnight. |
| **WindGustSpeed** | The speed of the strongest wind gust in the 24 hours to midnight. |
| **WindDir9am** | The direction of the strongest wind gust at 9 am. |
| **WindDir3pm** | The direction of the strongest wind gust at 3 pm. |
| **WindSpeed9am** | The speed of the strongest wind gust at 9 am. |
| **WindSpeed3pm** | The speed of the strongest wind gust at 3 pm. |
| **Humidity9am** | Humidity at 9 am. |
| **Humidity3pm** | Humidity at 3 pm. |
| **Pressure9am** | Atmospheric pressure at 9 am. |
| **Pressure3pm** | Atmospheric pressure at 3 pm. |
| **Cloud9am** | Cloud level at 9 am. |
| **Cloud3pm** | Cloud level at 3 pm. |
| **Temp9am** | Temperature at 9 am. |
| **Temp3pm** | Temperature at 3 pm. |
| **RainToday** | Whether raining today. |
| **RainTomorrow (Class Attribute)** | Whether raining tomorrow. |

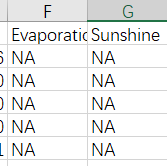
1. **Data Preparation and Preprocessing**

At the first time we load the source data file “**weatherAUS.csv**” into WEKA, we encountered three major problems.

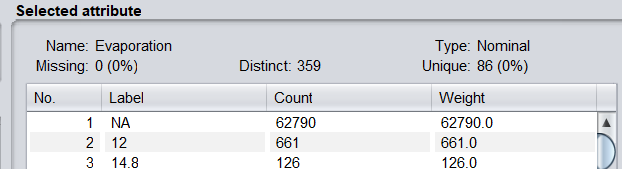
1. Attribute **MinTemp, MaxTemp, Rainfall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Cloud9am, Temp9am and Temp3pm** (**13** attributes in total) cannot be recognized by WEKA, as the picture below shows.

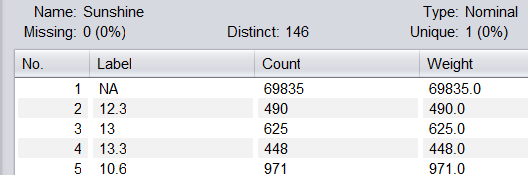


1. There are many missing values in the source data file “**weatherAUS.csv**”, picture below presents an example. “**NA**” means missing value.



However, WEKA cannot detect those missing values, but mistakes them as a valid value.

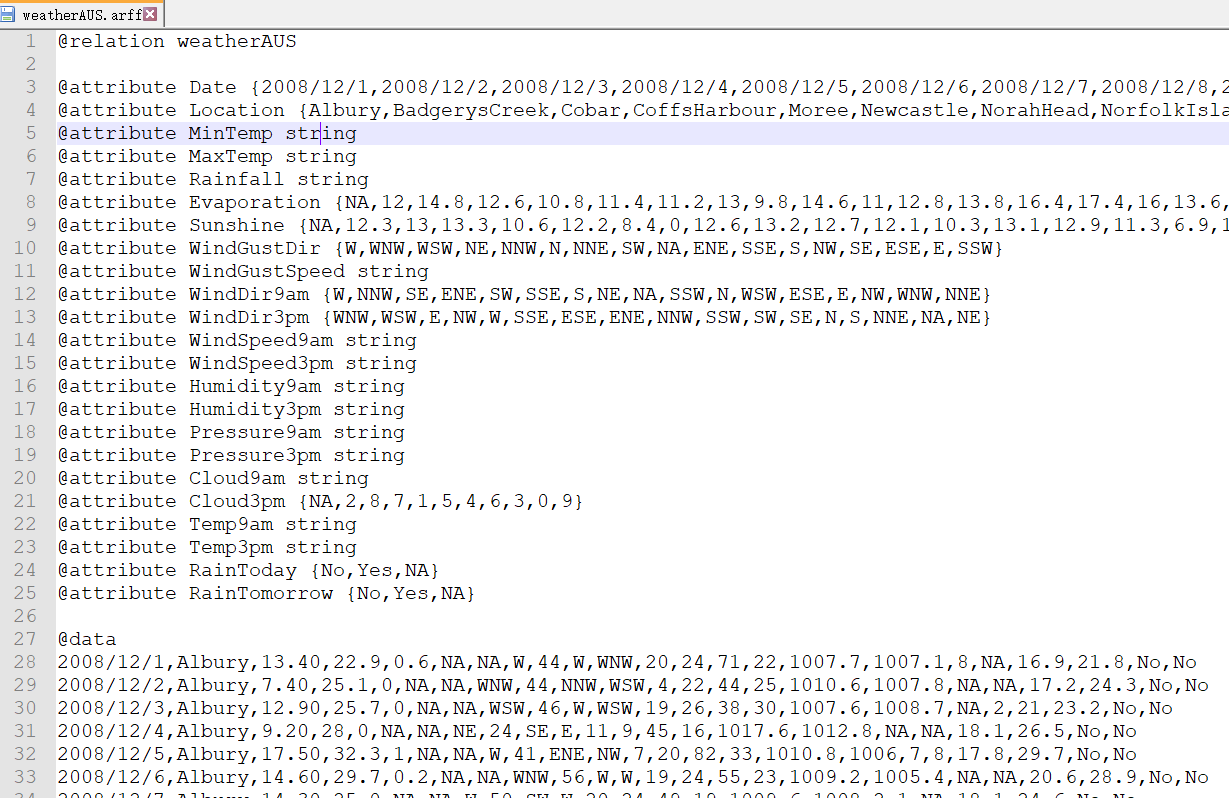




1. WEKA mistakes the **Could3pm** as nominal attribute.

With careful inspection, we found that this is because WEKA only recognize attributes from “**weatherAUS.csv**” as either “**Nominal**” or “**String**”, and all the “**String**” type attributes cannot be shown with concrete data in WEKA.

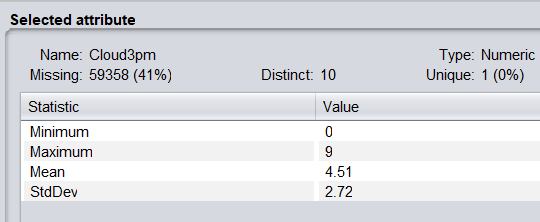
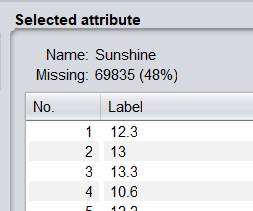
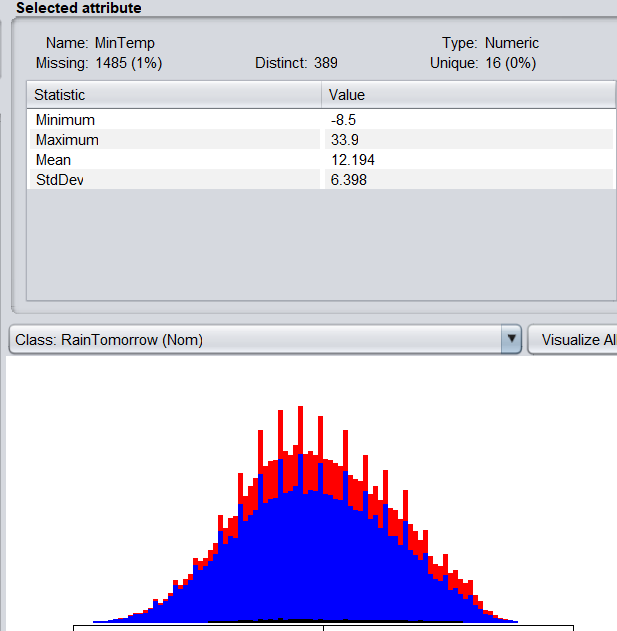
We decide to load the “**weatherAUS.csv**” file into WEKA, and “**save**” it as “**weatherAUS.arff**”. The arff file looks like this.



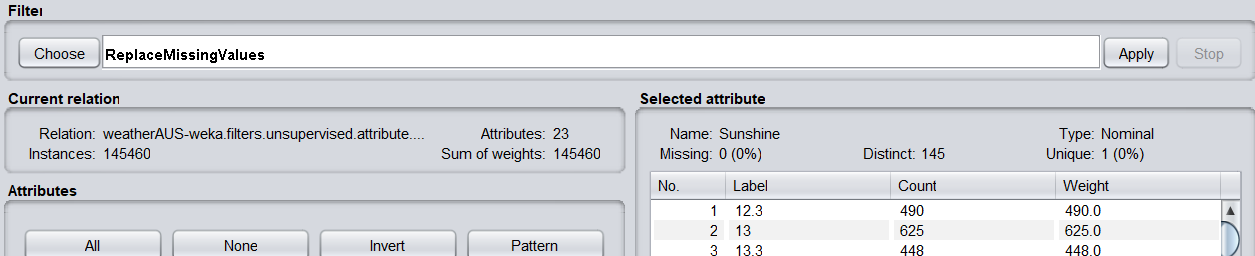
To tackle the three problems presented above, we

1. manually modify the “**string**” type with “**numeric**”, and remove the “**NA**” in the set of nominal attributes.
2. Replace all the “**NA**” in tuples with “**?**”, which is the standard representation of missing values in WEKA.
3. Modify the type of **Could3pm** from “**nominal**” to “**numeric**”.

Then, we reload “**weatherAUS.arff**” in WEKA, and found the three problems no longer exist, as the pictures below show.



Finally, we replace all the missing values using the method “**weka.filters.unsupervised.attribute.ReplaceMissingValues**”. As the picture below shows, all the missing values disappear.



Further, there are missing values in the class attribute “**RainTomorrow**” which cannot be filled in automatically, so we use the “**Edit**” function in WEKA to delete corresponding tuples manually.

1. **Attribute Selection and Data Mining Algorithms**

**Attribute Selection methods and results**

|  |  |
| --- | --- |
| **Attribute Selection Method** | **Selected Attributes** |
| **CfsSubsetEval from WEKA** |  |
| **InfoGainAttributeEval from WEKA** |  |
| **Manually Selection by ourself** | Rainfall, Humidity3pm, Cloud3pm, RainToday |

**Data Mining Algorithms**

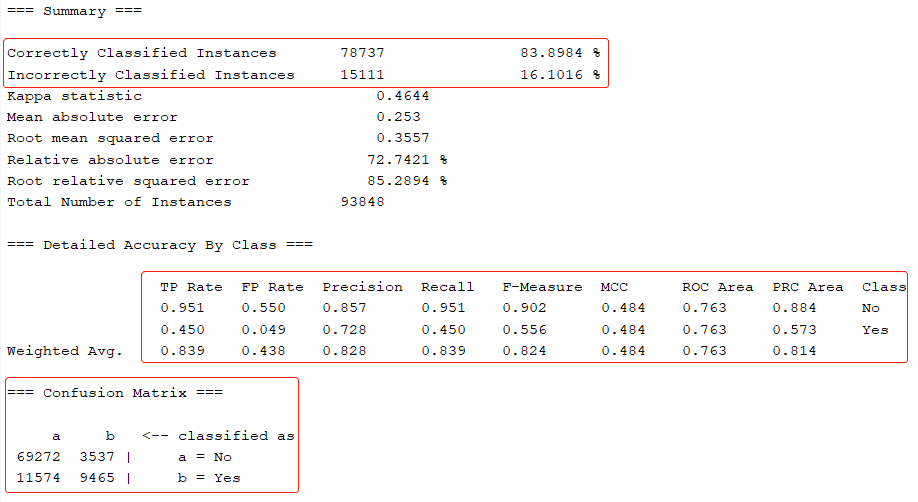
|  |  |
| --- | --- |
| **Algorithm** | **Description** |
| **J48** | J48 algorithm can manage numerical values, large data quantities, and datasets with missing values. |
| **RandomForest** | Random forest algorithm, the classification process uses more than one “tree”. Each tree produces a classifier, and these classifiers vote to determine the algorithm that gets the most votes. This classification algorithm is then used to classify the dataset. |
| **NaiveBayes** | Naive Bayes algorithm is based on the Bayesian theorem and operates on conditional probability, it is a powerful algorithm for predictive modeling. Additionally, the Naive Bayes classifier works quite well concerning real-world situations. |

**Data Set**

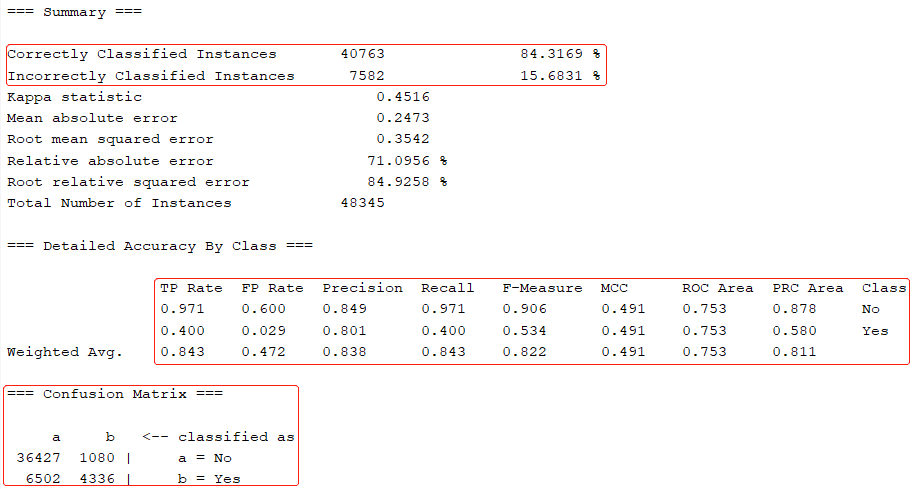
|  |  |  |  |
| --- | --- | --- | --- |
| **Data Set Name** | **Number of tuples** | **Number of negative tuples** | **Number of positive tuples** |
| **Total Set** | 142193 | 110316 | 31877 |
| **Training Set** | 93848 | 72809 | 21039 |
| **Test Set** | 48345 | 37507 | 10838 |

The training and test set are generated using the “**Edit**” function in WEKA. First, we sort the tuples according to class instance, from “No” to “Yes”; Then, we extract the first 72809 tuples as the negative training data and tuples within the position 110317 (inclusive) and 131355 (inclusive) as positive training data, the other data are stored as test set.

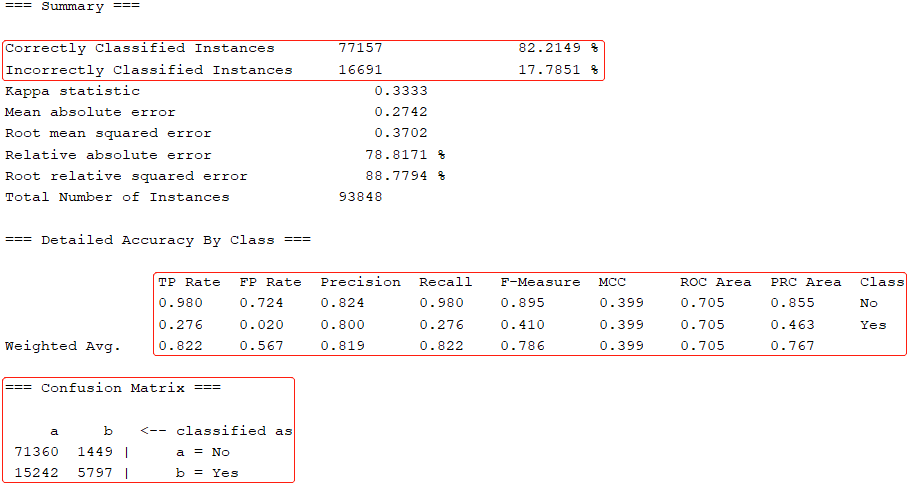
1. **CfsSubsetEval + J48**
   1. **Train**



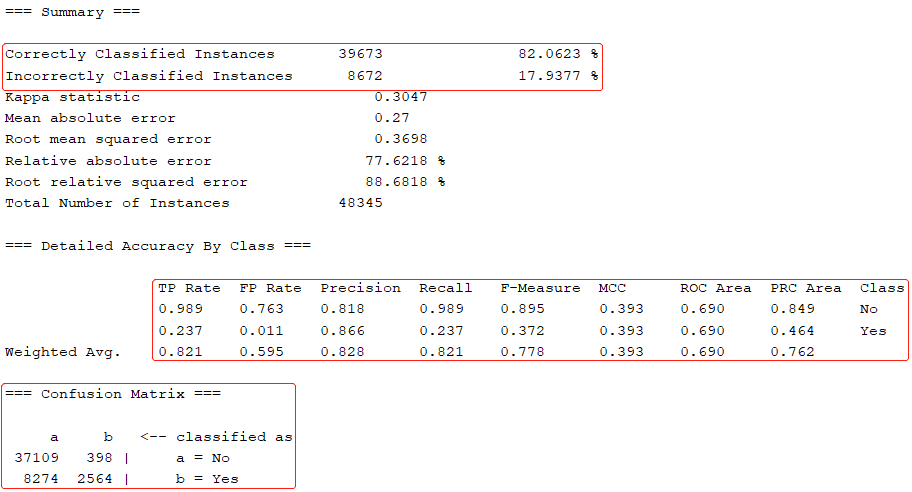
* 1. **Test**



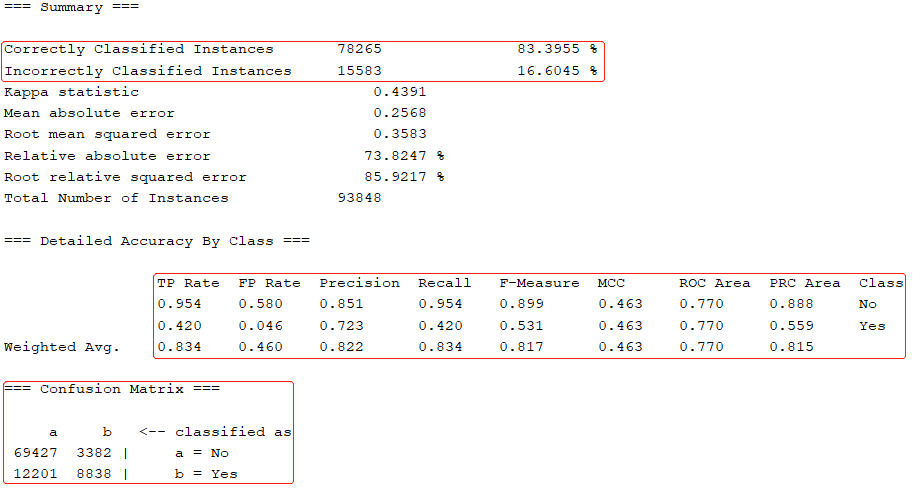
1. **InfoGainAttributeEval + J48**
   1. **Train**



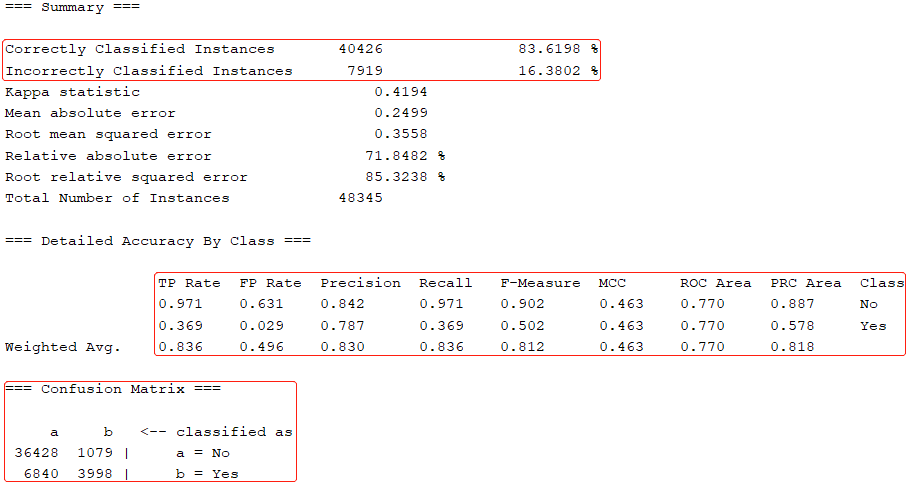
* 1. **Test**



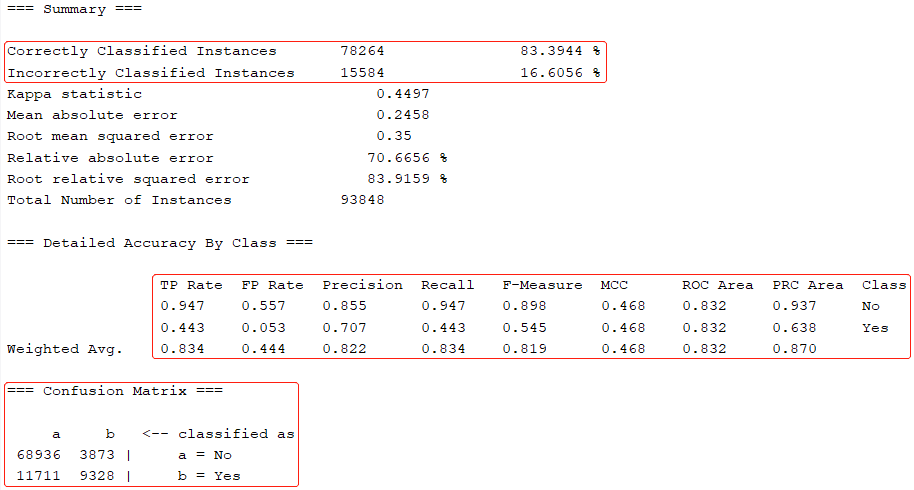
1. **Manual + J48**
   1. **Train**



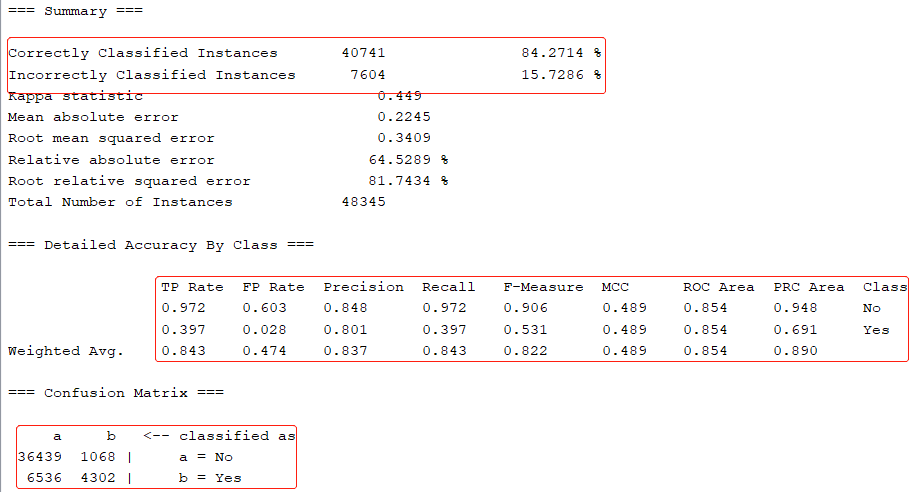
* 1. **Test**



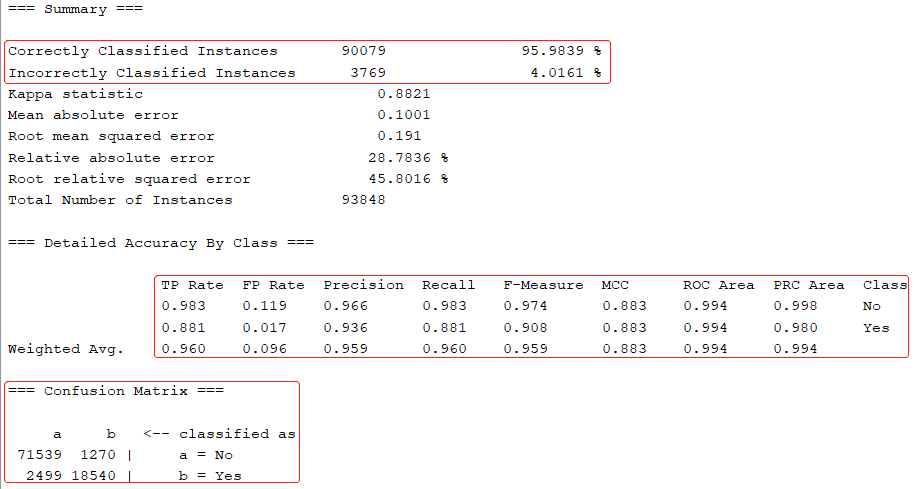
1. **CfsSubsetEval + RandomForest**
   1. **Train**



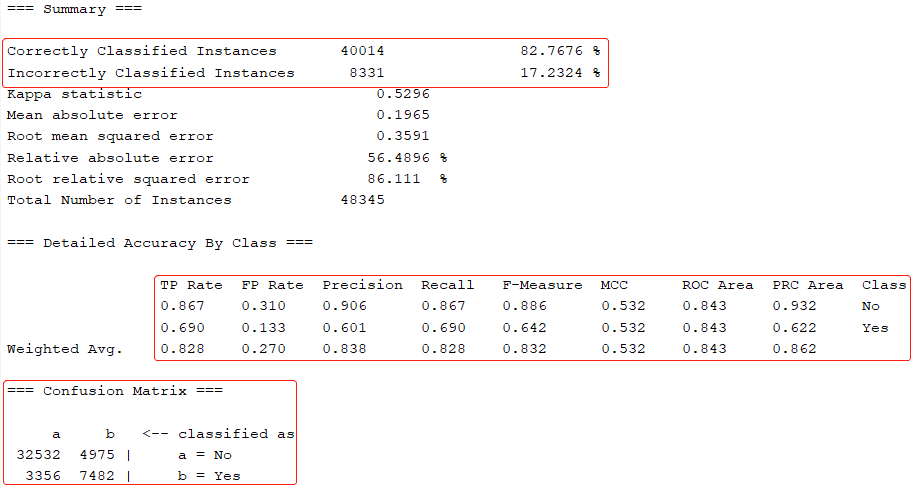
* 1. **Test**



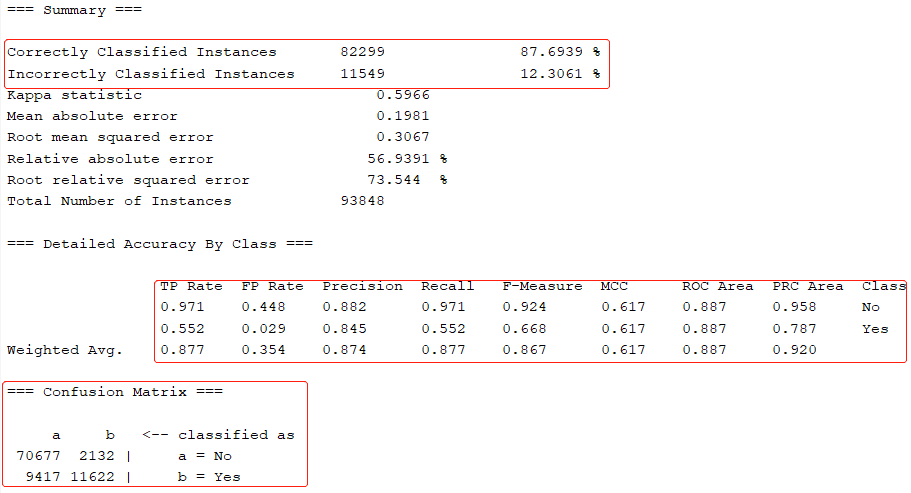
1. **InfoGainAttributeEval + RandomForest**
   1. **Train**



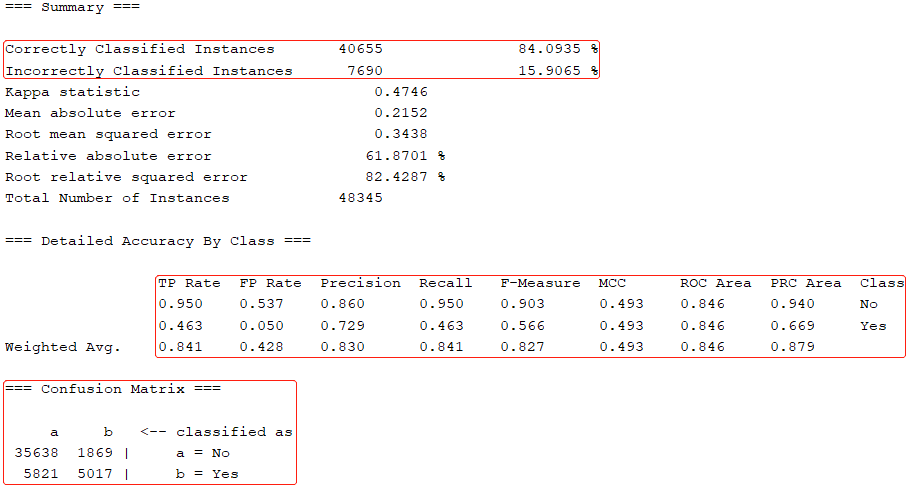
* 1. **Test**



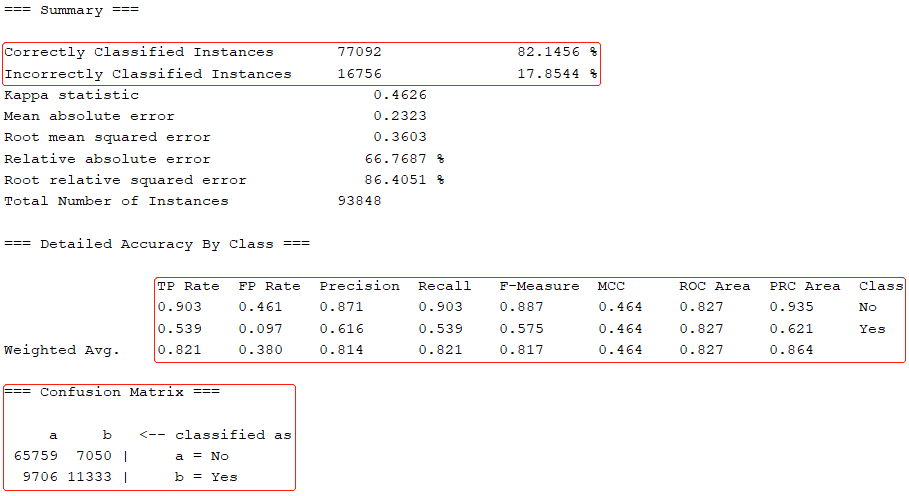
1. **Manual + RandomForest**
   1. **Train**



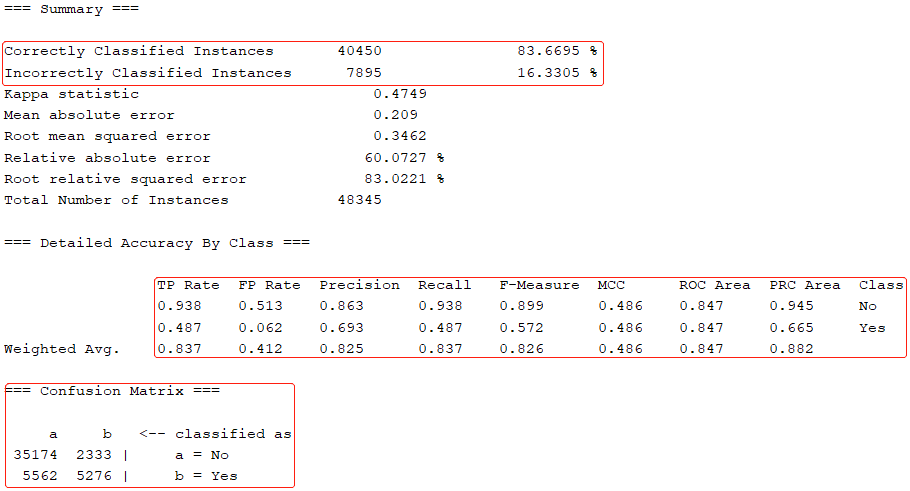
* 1. **Test**



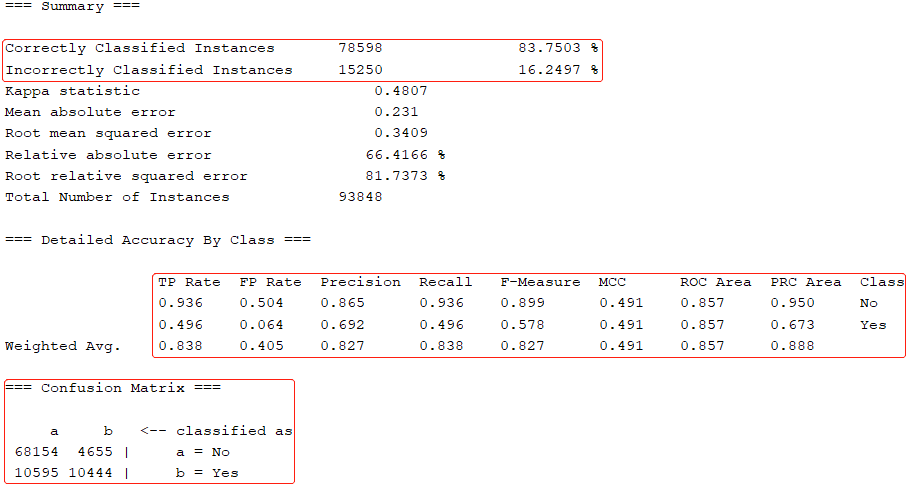
1. **CfsSubsetEval + NaiveBayes**
   1. **Train**



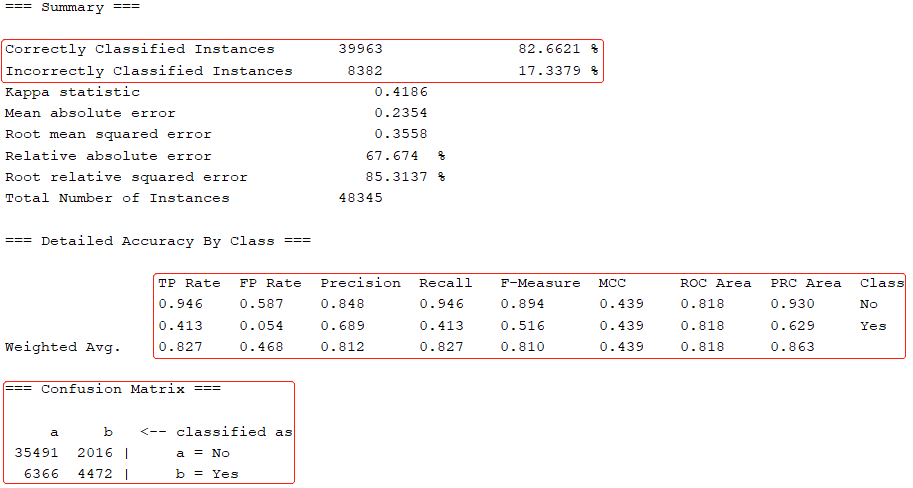
* 1. **Test**



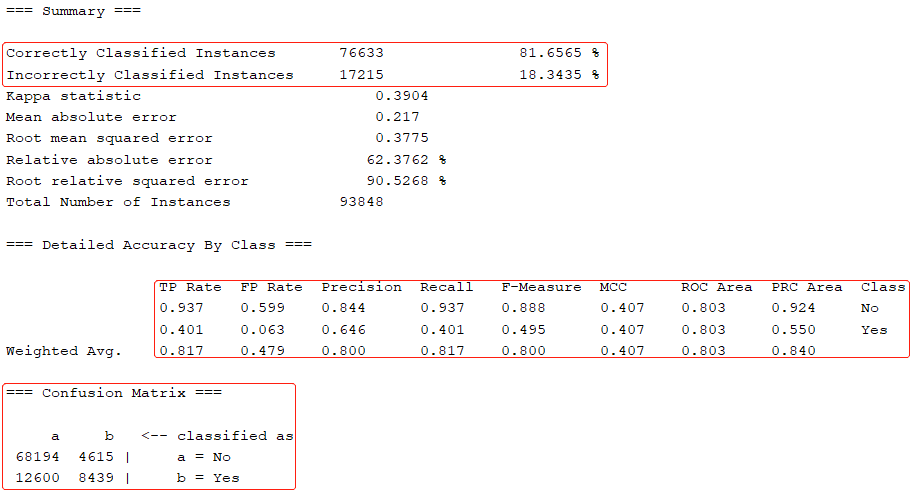
1. **InfoGainAttributeEval + NaiveBayes**
   1. **Train**



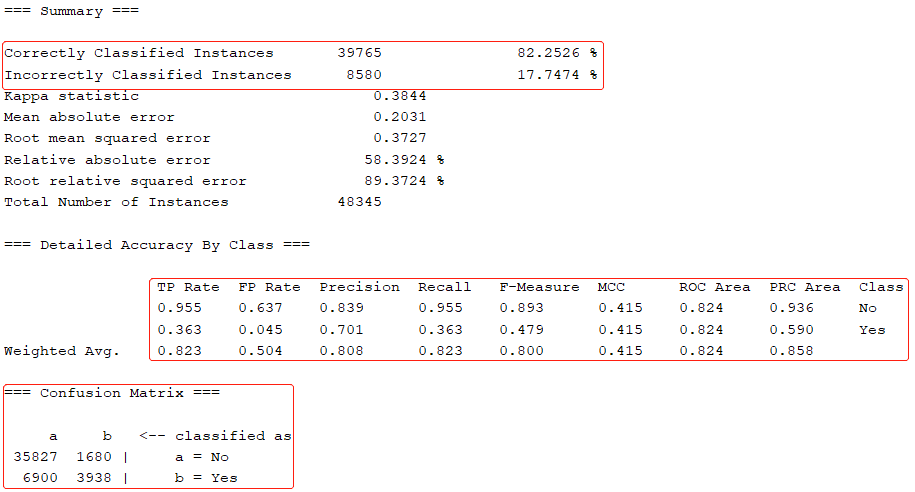
* 1. **Test**



1. **Manual + NaiveBayes**
   1. **Train**



* 1. **Test**



1. **Best Model**

From the above results, I choose random forest as the best model for my task, since it achieves slightly better performance in all measures.

1. **Discussion and Conclusion**

During the experiments, I encounter some practical problems, for example, I tended to use the “PrincipalComponents” method for attribute selection, but an out-of-memory occurs and WEKA exit immediately, and when I kept two many attributes, random forest also caused the out-of-memory problem. Thus, I have to choose other lightweight methods for attribute selection and keep the most important attribute for training and prediction.

There is no “the one” algorithm that is suitable for every task, and not all collected attributes are useful, obtaining a practical model requires both good attributes and good algorithms.