CSE477(1)-Project Report

Comparative analysis of different classification algorithms

Submitted to:

Dr. Mohammad Rezwanul Huq
Assistant Professor, Dept. of CSE, EWU

Submitted by:

Anamika Das Mou (2015-2-60-080)

Introduction

In modern life, Data have increased rapidly and using those data we can measure the prediction and we also can make a proper decision to execute some works where result may use for business purpose or may be used for disease or other sectors. From the large number of dataset, we can find patterns and patterns help to make decisions. The data analysis occurs by data mining. The Data mining is the process of sorting through large data sets to identify patterns and establish relationships to solve problems through data analysis. Data mining tools allow enterprises to predict future trends.

Here, we worked with one dataset. For the python implementation, we had used dataset and for the Weka tools we had also used the same dataset.
Dataset**[who_suicide.csv]** about suicide information, here have 6 attributes. Using those attributes, we have to measure in which range age the suicide will be occurred.

For this dataset we used Naïve Bayes Classifier, Decision Tree Classifier, k Nearest-Neighbour Classifier, Random Forest tree classifier. We also used the same algorithms for Weka tools.

Classification Algorithm

We have used four algorithms to measure the result.
1. Decision Tree
2. Naïve Bayes
3. Random forest
4. KNN

Decision Tree: A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision trees
are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. This algorithm has the ability to handle binary and multiclass classification problem. We need two different entropies for calculating.one is for total database that means for all the attribute and the equation for that is,
Info(F) = $-\sum_{i=1}^{m} p_i \log_2(p_i)$
pi is the probability that an arbitrary tuple in F belongs to class Yes or No and another is for each attribute and equation for that is,

$\text{Info}_{att}(F) = \sum |Fj|$
$v\ j=1\ |F| \times info(F)$
After that total gain will be calculated. Here, the formula,
$\text{Gain(attribute)} = \text{Info}(F) - \text{Info}_{att}(F)$
Naïve Bayes: In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters
linear in the number of variables (features/predictors) in a learning problem.
Maximum-likelihood
training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.
By using the Bayes theorem, posterior probability can be calculated, P(c|x), from P(c), P(x), and P(x|c). The Naive Bayes classification consider the predictor value (x) on a given category (c) is independent which is conditional independence. So, we can formulate the theorem from Bayesian.
Here,
$P(c|x_i) = p(x_1|c) \text{ x } \dots \text{ x } P(x_n|c) \text{ x } P(c)$
P(c|xi) is the posterior probability of a given predictor (attribute). P(c) is the prior probability of class. P(xi|c) is the likelihood. P(xi) is the prior probability of the predictor.
Random forest:

Random forests or random decision forests are an ensemble learning method for
classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. To say it in simple words: Random forest builds multiple
decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems.
Random Forest has nearly the same hyper parameters as a decision tree or a bagging classifier. Fortunately, we don't have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Like I already said, with Random Forest, you can also
deal with Regression tasks by using the Random Forest repressor.
Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better
model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. We can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds like a normal decision tree does. This is:
1. Increasing the Predictive Power.
2. Increasing the Model's Speed.

k Nearest-Neighbour Classifier:

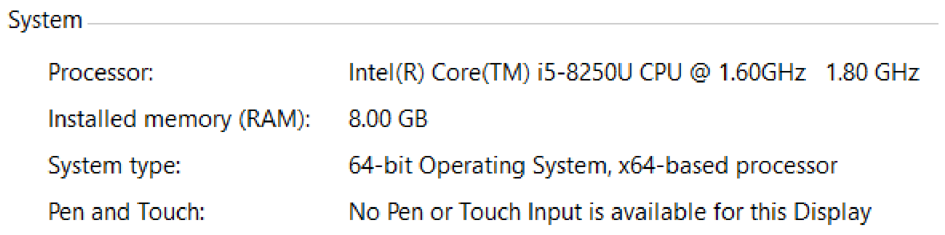
K-Nearest Neighbors is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use a data and classify new data points based on a similarity measures. It works based on minimum distance from the query instance to the training samples to determine the K-nearest neighbors. The data for KNN algorithm consist of several multivariate attributes name that will be used to classify. KNN is a lazy learning algorithm.

Dataset

Firstly, we had chosen the **"who_suicide"** dataset for the python implementation and also for Weka tools. Here have
6 attributes and it's with many null values. In this dataset, there are 43777 data. This dataset is a suicide information list. The list has briefly shows about the suicide tendency. At the end this data help to
predict what range age is particularly suicidal.

Implementation

The configuration of my Pc is:


System

Processor:	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Installed memory (RAM):	8.00 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Weka Tools: In below, there have all attribute figures.
In the weka tool, we have used 75% data for the training and 25% data for the testing set. So in below we attached different method measures to observe the different performance.

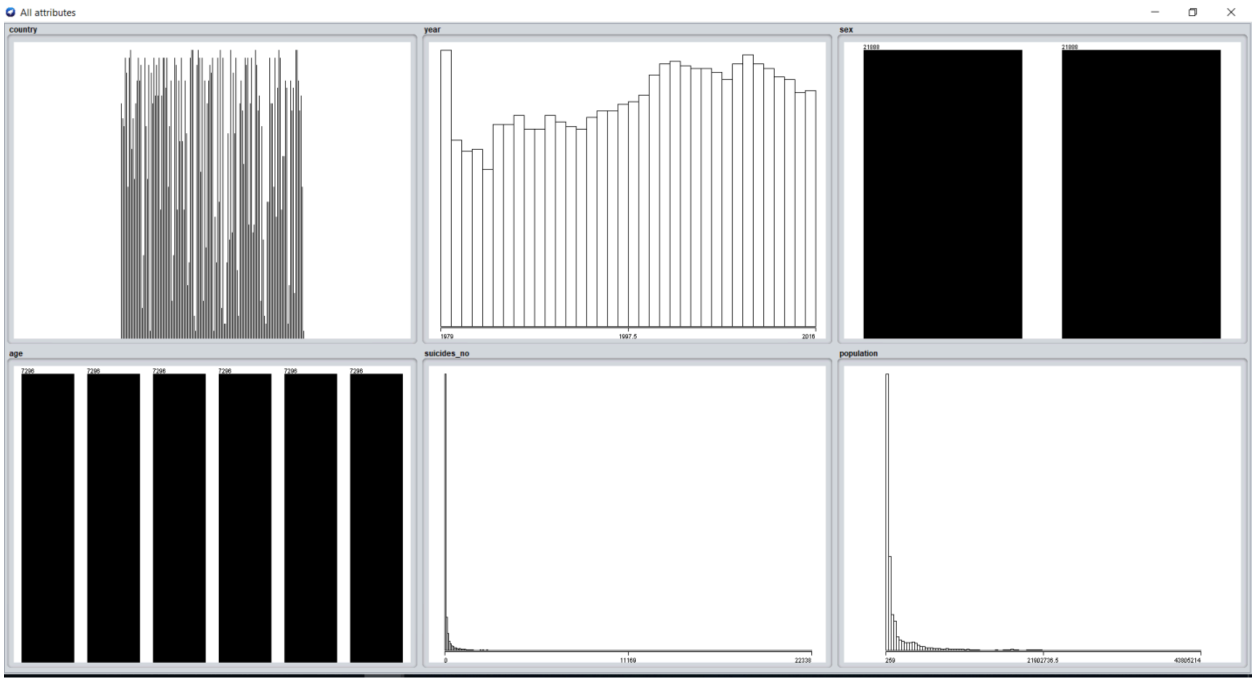
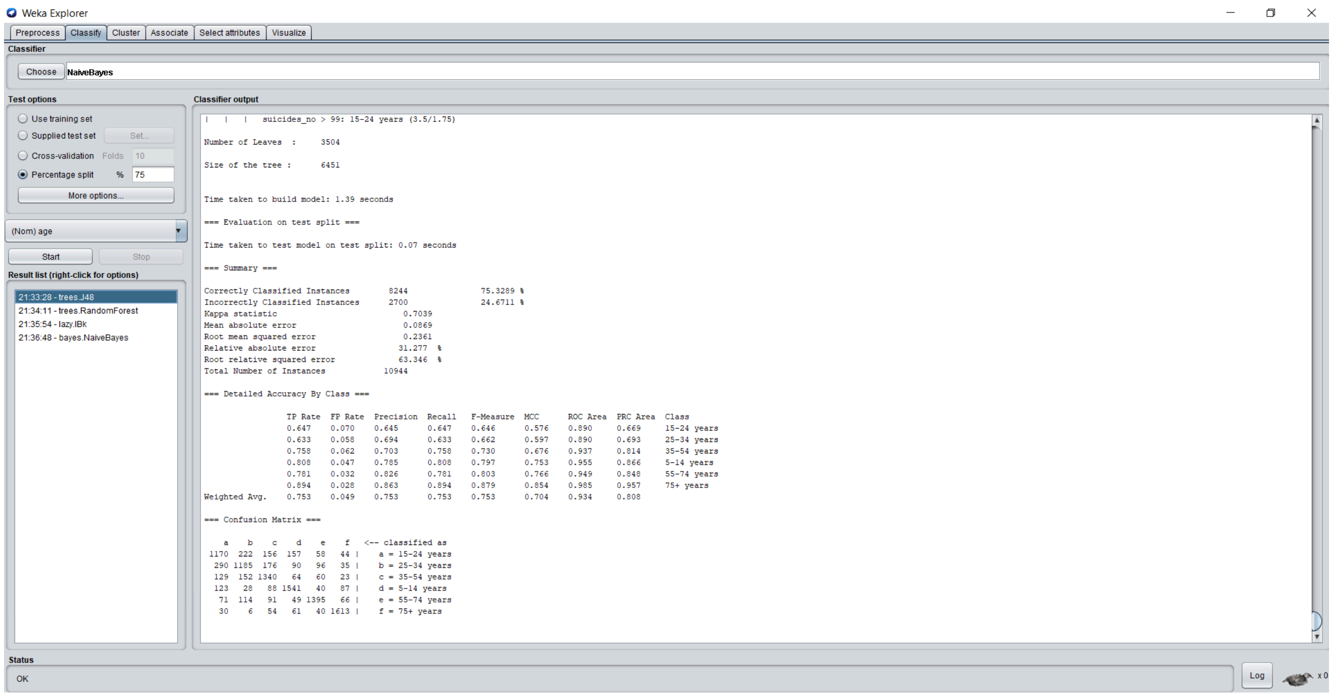


Figure1:- All attributes chart

Decision Tree: Correctly classified instances are 8244, about 75.31% and incorrectly classified instances are 2700, about 24.67% among the total number of 10966 instances. In Decision Tree algorithm the accuracy for that dataset is about 75%



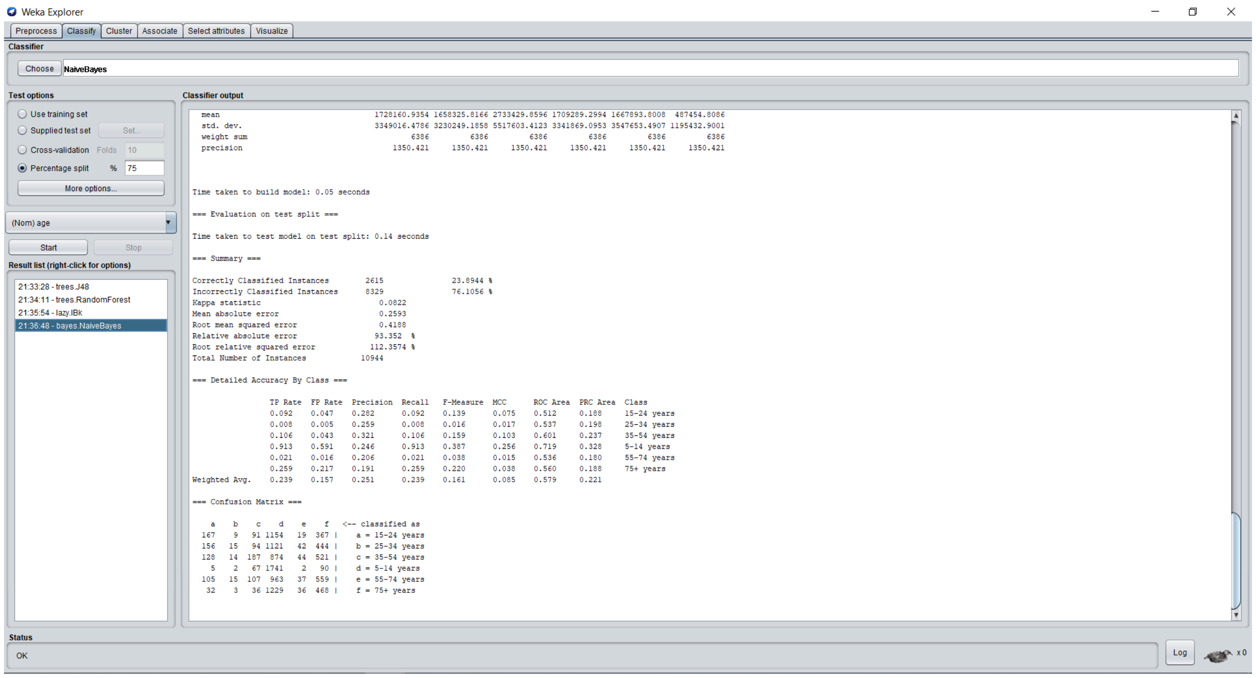
Classifier: NaiveBayes

Test options: Cross-validation Folds 10; [✓ Percentage split] % 75

(Nom) age

```
|   |   |   suicides_no > 99: 15-24 years (3.5/1.75)

Number of Leaves  :     3504

Size of the tree :      6451


Time taken to build model: 1.39 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.07 seconds

=== Summary ===

Correctly Classified Instances        8244               75.3289 %
Incorrectly Classified Instances      2700               24.6711 %
Kappa statistic                          0.7039
Mean absolute error                      0.0869
Root mean squared error                  0.2361
Relative absolute error                 31.277  %
Root relative squared error             63.346  %
Total Number of Instances            10944

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.647    0.070    0.645      0.647    0.646      0.576    0.890     0.669     15-24 years
                 0.633    0.058    0.694      0.633    0.662      0.597    0.890     0.693     25-34 years
                 0.758    0.062    0.703      0.758    0.730      0.676    0.937     0.814     35-54 years
                 0.808    0.047    0.785      0.808    0.797      0.753    0.955     0.866     5-14 years
                 0.781    0.032    0.826      0.781    0.803      0.766    0.949     0.848     55-74 years
                 0.894    0.028    0.863      0.894    0.879      0.854    0.985     0.957     75+ years
Weighted Avg.    0.753    0.049    0.753      0.753    0.753      0.704    0.934     0.808

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1170  222  156  157   58   44 |    a = 15-24 years
  290 1185  176   90   96   35 |    b = 25-34 years
  129  152 1340   64   60   23 |    c = 35-54 years
  123   28   88 1541   40   87 |    d = 5-14 years
   71  114   91   49 1395   66 |    e = 55-74 years
   30    6   54   61   40 1613 |    f = 75+ years
```

Naïve Bayes: Correctly classified instances are 2615, 23.89% and incorrectly classified instances are 8329, 76.1% among the total number of 10944 instances. In Naïve Bayes algorithm the accuracy for that dataset is about 23.89%



Choose: NaiveBayes

Cross-validation Folds: 10

[✓ Percentage split] %: 75

(Nom) age

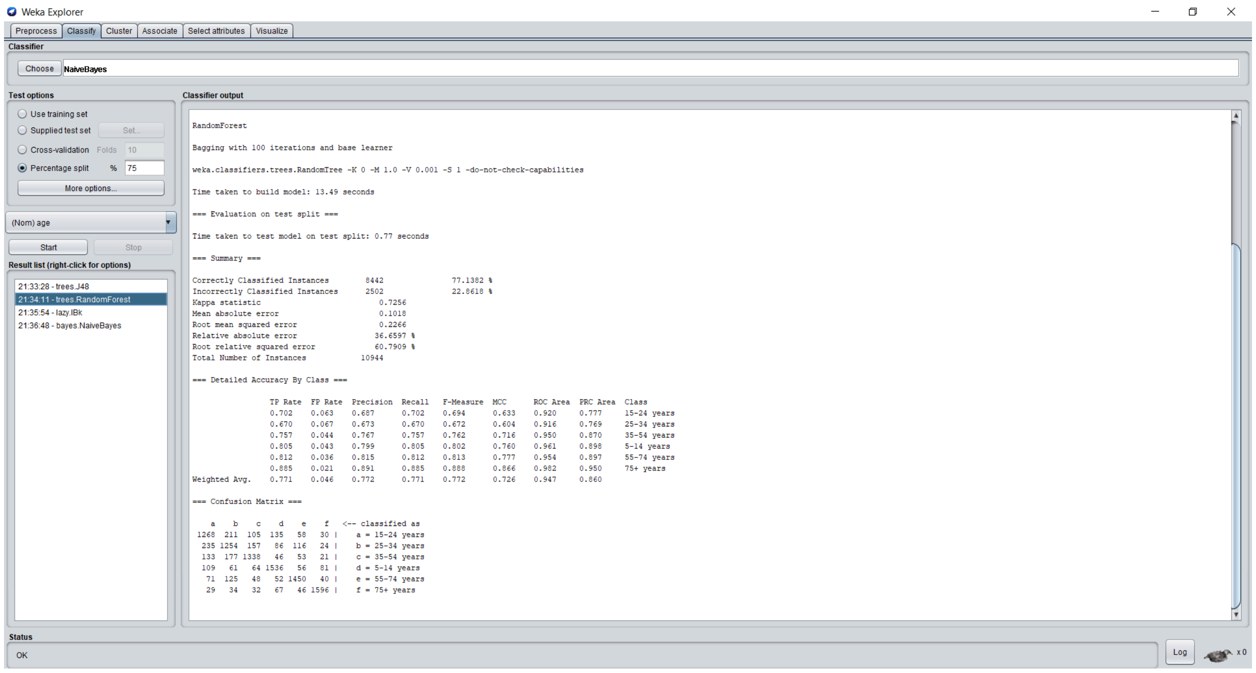
Result list:
21:33:28 - trees.J48
21:34:11 - trees.RandomForest
21:35:54 - lazy.IBk
21:36:48 - bayes.NaiveBayes

Status: OK

```
  mean                          1728160.9354 1658325.8166 2733429.8596 1709289.2994 1667893.8008  487454.8086
  std. dev.                     3349016.4786 3230249.1858 5517603.4123 3341869.0953 3547653.4907 1195432.9001
  weight sum                            6386         6386         6386         6386         6386         6386
  precision                         1350.421     1350.421     1350.421     1350.421     1350.421     1350.421



Time taken to build model: 0.05 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.14 seconds

=== Summary ===

Correctly Classified Instances        2615               23.8944 %
Incorrectly Classified Instances      8329               76.1056 %
Kappa statistic                          0.0822
Mean absolute error                      0.2593
Root mean squared error                  0.4188
Relative absolute error                 93.352  %
Root relative squared error            112.3574 %
Total Number of Instances            10944

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.092    0.047    0.282      0.092    0.139      0.075    0.512     0.188     15-24 years
                 0.008    0.005    0.259      0.008    0.016      0.017    0.537     0.198     25-34 years
                 0.106    0.043    0.321      0.106    0.159      0.103    0.601     0.237     35-54 years
                 0.913    0.591    0.246      0.913    0.387      0.256    0.719     0.328     5-14 years
                 0.021    0.016    0.206      0.021    0.038      0.015    0.536     0.180     55-74 years
                 0.259    0.217    0.191      0.259    0.220      0.038    0.560     0.188     75+ years
Weighted Avg.    0.239    0.157    0.251      0.239    0.161      0.085    0.579     0.221

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  167    9   91 1154   19  367 |    a = 15-24 years
  156   15   94 1121   42  444 |    b = 25-34 years
  128   14  187  874   44  521 |    c = 35-54 years
    5    2   67 1741    2   90 |    d = 5-14 years
  105   15  107  963   37  559 |    e = 55-74 years
   32    3   36 1229   36  468 |    f = 75+ years
```

Figure3:- Naïve Bayes observation

Random forest: Correctly classified instances are 8442, 77.1% and incorrectly classified instances are 2502, 22.86% among the total number of 10944 instances. In Random forest algorithm
the accuracy for that dataset is about 77%



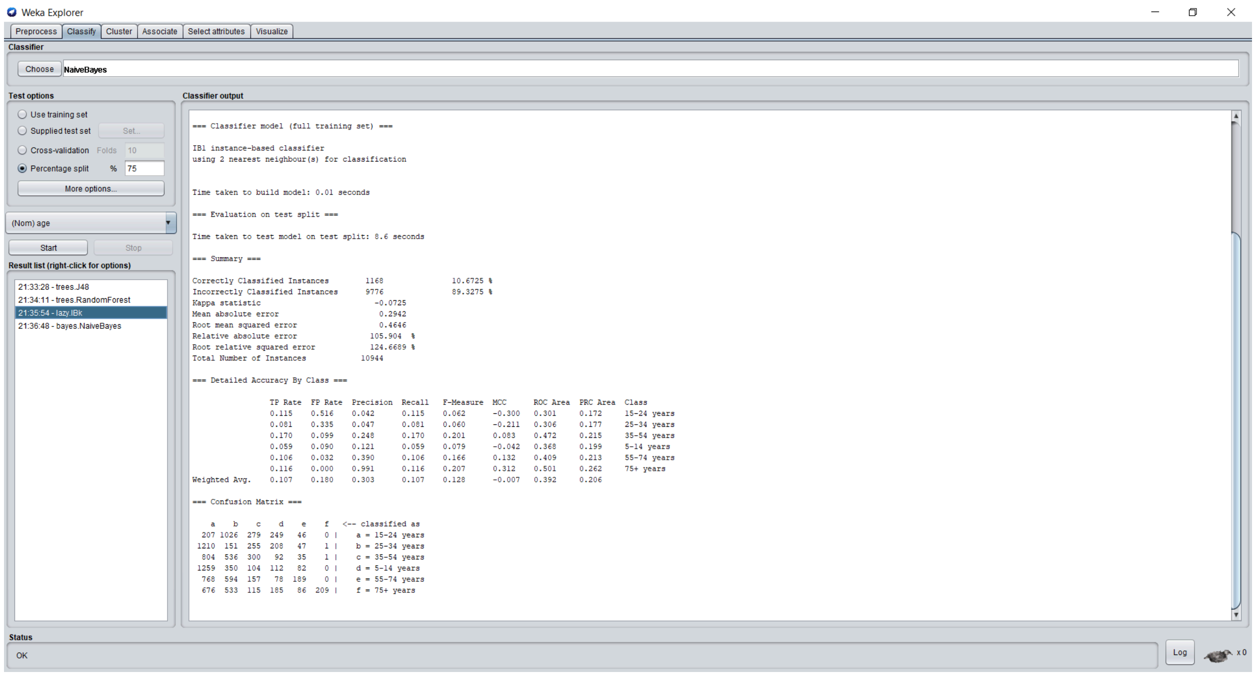
```
Classifier: NaiveBayes
Test options: Cross-validation  Folds 10
Percentage split  %  75
(Nom) age

Result list (right-click for options)
21:33:28 - trees.J48
21:34:11 - trees.RandomForest
21:35:54 - lazy.IBk
21:36:48 - bayes.NaiveBayes

Status
OK
```

```
RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 13.49 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.77 seconds

=== Summary ===

Correctly Classified Instances        8442               77.1382 %
Incorrectly Classified Instances      2502               22.8618 %
Kappa statistic                          0.7256
Mean absolute error                      0.1018
Root mean squared error                  0.2266
Relative absolute error                 36.6597 %
Root relative squared error             60.7909 %
Total Number of Instances            10944

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.702    0.063    0.687      0.702    0.694      0.633    0.920     0.777     15-24 years
                 0.670    0.067    0.673      0.670    0.672      0.604    0.916     0.769     25-34 years
                 0.757    0.044    0.767      0.757    0.762      0.716    0.950     0.870     35-54 years
                 0.805    0.043    0.799      0.805    0.802      0.760    0.961     0.898     5-14 years
                 0.812    0.036    0.815      0.812    0.813      0.777    0.954     0.897     55-74 years
                 0.885    0.021    0.891      0.885    0.888      0.866    0.982     0.950     75+ years
Weighted Avg.    0.771    0.046    0.772      0.771    0.772      0.726    0.947     0.860

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1268  211  105  135   58   30 |    a = 15-24 years
  235 1254  157   86  116   24 |    b = 25-34 years
  133  177 1338   46   53   21 |    c = 35-54 years
  109   61   64 1536   56   81 |    d = 5-14 years
   71  125   48   52 1450   40 |    e = 55-74 years
   29   34   32   67   46 1596 |    f = 75+ years
```

Figure4:- Random forest observation

KNN:

Correctly classified instances are 1164, 10.67% and incorrectly classified instances are 9776, 89.33% among the total number of 10944 instances. In KNN algorithm the accuracy for that dataset is about 10.97%



```
Classifier: NaiveBayes
Test options: Cross-validation  Folds 10
Percentage split  %  75
(Nom) age

Result list (right-click for options)
21:33:28 - trees.J48
21:34:11 - trees.RandomForest
21:35:54 - lazy.IBk
21:36:48 - bayes.NaiveBayes

Status
OK
```

```
=== Classifier model (full training set) ===

IB1 instance-based classifier
using 2 nearest neighbour(s) for classification


Time taken to build model: 0.01 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 8.6 seconds

=== Summary ===

Correctly Classified Instances        1168               10.6725 %
Incorrectly Classified Instances      9776               89.3275 %
Kappa statistic                         -0.0725
Mean absolute error                      0.2942
Root mean squared error                  0.4646
Relative absolute error                105.904  %
Root relative squared error            124.6689 %
Total Number of Instances            10944

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.115    0.516    0.042      0.115    0.062      -0.300   0.301     0.172     15-24 years
                 0.081    0.335    0.047      0.081    0.060      -0.211   0.306     0.177     25-34 years
                 0.170    0.099    0.248      0.170    0.201      0.083    0.472     0.215     35-54 years
                 0.059    0.090    0.121      0.059    0.079      -0.042   0.368     0.199     5-14 years
                 0.106    0.032    0.390      0.106    0.166      0.132    0.409     0.213     55-74 years
                 0.116    0.000    0.991      0.116    0.207      0.312    0.501     0.262     75+ years
Weighted Avg.    0.107    0.180    0.303      0.107    0.128      -0.007   0.392     0.206

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  207 1026  279  249   46    0 |    a = 15-24 years
 1210  151  255  208   47    1 |    b = 25-34 years
  804  536  300   92   35    1 |    c = 35-54 years
 1259  350  104  112   82    0 |    d = 5-14 years
  768  594  157   78  189    0 |    e = 55-74 years
  676  533  115  185   86  209 |    f = 75+ years
```

Figure4:- KNN observation

Performance Evolution

In **table-1** in this section we show about weka tool implemented accuracy.

Algorithm	Precision	Recall	f-score	Support	Accuracy
Decision Tree	.645	.647	.646	10944	75%
Naïve Bayes	.282	.092	.139	10944	23.89%
KNN	.042	.115	.062	10944	10.97%
Random Forest	.687	.702	.694	10944	77%

Table1: Weka tool Implemented Confusion Matrix Report

In **table-2** we have shown python implemented accuracy

Algorithm	Precision	Recall	f-score	Support	Accuracy
Decision Tree	1	1	1	1513	100%
Naïve Bayes	.2	.3	.2	1513	20.31%
KNN	.36	.20	.26	2709	22.85%
Random Forest	1	1	1	1513	100%

Discussion

In table-1 we have seen Precision, Recall, f-score, Support, Accuracy. In table-1 shows Random forest gives the best result. For the increase of data, Random forest is a good classifier. So in this project random forest gives the best data for less data.
Decision tree and Random forest both give the best result for table-2 .In table-2, Decision tree and Random Forest give the same accuracy.

Conclusion

We learned some essentials about decisions trees, Naïve Bayes, random forests, KNN. We applied those algorithms to a dataset, and we compared the accuracy of those models. So all of the algorithms help to make decisions or predict. Using those algorithms, businessmen can predict their business and the mining might bring out the maximum profit.