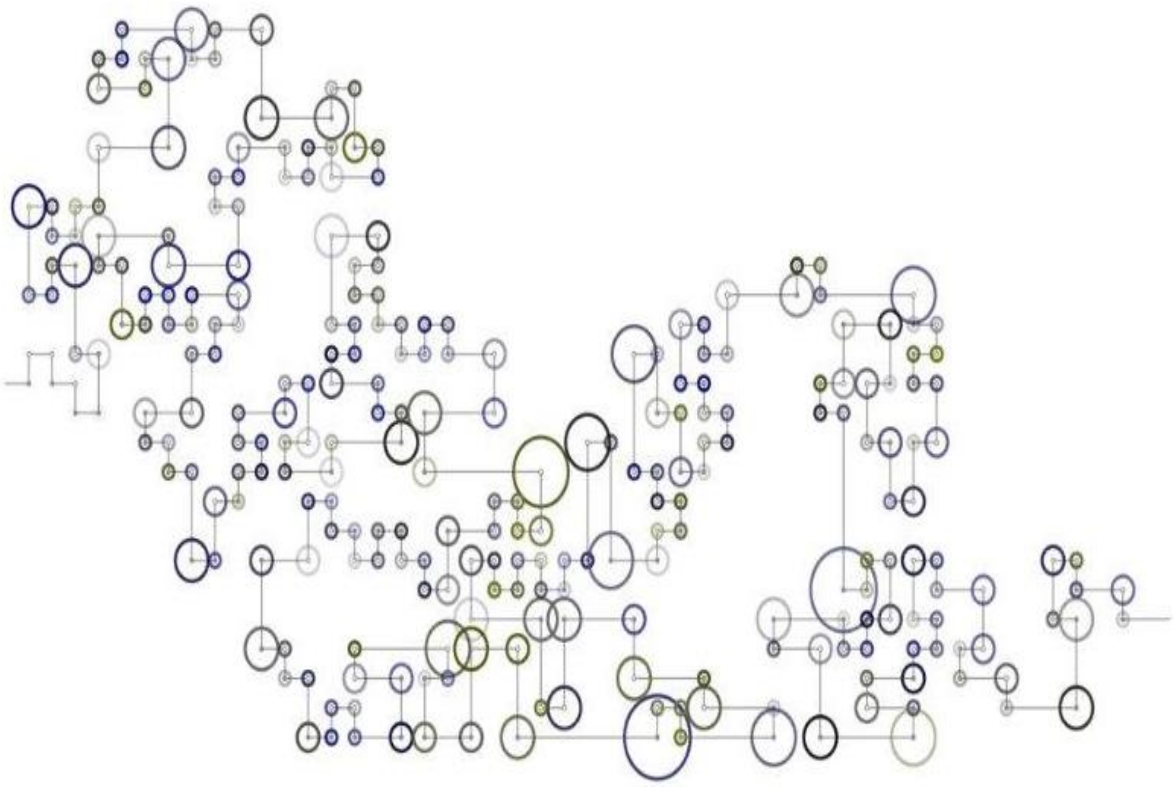


Use of cost-sensitive algorithms in Online Marketing

MSc in Data Science

Project Dissertation

Academic year: 2018/19

Author: Carlos Beltrán

Supervisor: Dr. Eman Nashnush

ABSTRACT

"Use of cost-sensitive algorithms in Online Marketing" has attempted to introduce the most relevant literature in cost-sensitive algorithms field. Moreover, the project has consisted of performing several tests (Direct and indirect method) with different types of cost-sensitive algorithms in an Online Marketing Dataset that was unbalanced, where a customer may or may not buy. This project has attempted to mitigate the rare class problems of the dataset by deploying cost-sensitive algorithms and thus minimizing the costs involved when misclassification occurs.

The results of performing the different test with the cost-sensitive models were compared and analysed. It has proved that the cost-sensitive algorithms are not that effective on mitigating the rare class problem, at least in Marketing, except for the over-sampling method that performed well.

Further works could include optimization of algorithms parameters to see whether the parameters selected made the algorithms underperform and testing the algorithm in more Marketing datasets to assess the consistency of over-sampling across different marketing datasets.

ACKNOWLEDGEMENTS

I owe my sincere gratitude to my project supervisor Dr Eman Nashnush, who took a keen interest in the project work and guided me along, until the completion of the project work by providing all the necessary assistance and guidance.

CONTENTS

TABLE OF CONTENTS

1 INTRODUCTION

In the new era of digitalization is a must for businesses around the world to have an online presence. A website is not sufficient anymore to reach customers, and companies need to invest billions of pounds every year in digital marketing to sell online.

The traditional way to do digital marketing is a mix of analytics and intuition gained over the year by its practitioners. However, with the emergence of big data and data science, the digital marketing departments around the globe can take more sophisticated decision based on numeric and logical thinking rather than intuition. It is a game-changing factor in the industry, and the algorithms will play a significant role in the future of online sales, and therefore in the profitability of many companies.

Until now, most of the research has been done on algorithms that do not take into consideration the cost of the misclassification, and it is only concerned is to improve the accuracy in the model. It translated to the digital marketing industry means millions of pounds wasted targeting customers that were not interested in our products and services and wrong expectation on incomes coming from marketing. Therefore, there is a need to use a different approach to the problem which avoids companies to incur in such costs.

Insensitive algorithms are built to produce optimal results under certain conditions, such as having a balanced data structure. However, sales data is unbalanced; thus, we cannot expect insensitive algorithms to perform correctly and produces optimal results. For this reason, cost-sensitive algorithms which are built under the assumption of unbalanced data perform well on sales datasets and helps to reduce the misclassification, which it directly translates in money saved and a better forecast of sales.

This project aims to introduce cost-sensitive algorithms and the most relevant literature in the field. It will show the cost-sensitive algorithms use in the Online Marketing domain and the exploration and pre-possessing of an Online Marketing dataset. Besides, a comparison of the outcomes produced between the cost-sensitive algorithms and the insensitive algorithms. Finally, it will critically evaluate the results and the findings.

2 THE MOTIVATION FOR THE PROJECT

The motivation for the project is that most of the algorithms in use nowadays do not consider cost when they are built. In the case of Online Marketing, the cost of misclassification is a relevant factor, and overlooking it may lead to waste of resources and wrong business decisions.

Therefore, there is a need to deploy cost-sensitive algorithms in such a domain to improve Online Marketing conversions and as a result, improving businesses profitability.

3 THE AIMS AND OBJECTIVES

The aims and objectives of this projects are:

1- Introduction of cost-sensitive algorithms.

2- In-depth investigation of the various literature sources on cost-sensitive algorithms.

3- Deployment of cost-sensitive algorithms on an Online Marketing dataset.

4- Compare cost-sensitive algorithms under the cost viewpoint.

5- Critical evaluation of the results.

6- Areas for further research.

4 BACKGROUND

Typically, most of the algorithms are built to maximize correct classified cases. However, insensitive algorithms do not consider cost, and the assumption that all costs are the same is made. This assumption in many domains is irrelevant, but in other such as medicine, finance, or digital marketing, it does matters. Overlooking costs may lead to disastrous consequences. Below we will introduce the most relevant literature in cost-sensitive algorithms and its applications in marketing.

Turney [1][2] introduces on his extensive work, the concept of different costs involved, and the importance to avoid assuming that all costs are the same. Moreover, He introduces the idea of cost-sensitive algorithms and defines them intending to achieve high accuracy but minimizing cost. Firsts implementations on cost-sensitive are found on the academic literature [2][3][4] on medicine for making a diagnosis and in finance to detect fraud.

Since Insensitive algorithms are focused on maximizing accuracy and cost-sensitive is focused on minimizing costs, a few researchers started working on algorithms which could have bound the advantages of both together. Elkan [5] introduces the concept of optimal solution on cost-sensitive algorithms, which means minimizing the misclassifications, whereas accuracy is maximized.

Cost-sensitive algorithms are generally classed in the literature in two groups [6]. First is known as the direct method and built a cost-sensitive classifier, whereas the second group is known as an indirect method which is making an insensitive algorithm into cost-sensitive using an intermediate stage that involves creating a wrapper. These methods will be explained further in detail in the section 4.3.

The current state of the art of cost-sensitive algorithms on digital marketing is not extensive. The most relevant research was made Khor Kok-Chin, and Ng Keng-Hoong [7] in an experiment where an imbalanced bank direct marketing data set was used to test cost-sensitive algorithms to mitigate the problem with the cost of misclassification. The paper tested three cost-insensitive algorithms such as the Naive Bayes, C4.5 and Naive Bayes Tree, and they were compared with two cost-sensitive algorithms such as SVM and over-sampling method. The research found that cost-sensitive algorithms did not handle well-imbalanced datasets because they are not designed to handle imbalanced class distributions. Therefore, it may not work well for certain imbalanced data sets. On the other hand, Over-sampling, worked well for the dataset and helped to generalize the decision region of the rare class clearly and subsequently improved the classification result.

4.1 TYPE OF COST

Data mining is an interdisciplinary subfield of computer science and statists [8]. Data mining can be described as the use of efficient techniques for the analysis of vast collections of data and the extraction of useful and possibly unexpected patterns in data [9]. Data mining involves machine learning, statistics, and database systems techniques.

Cost-sensitive is a subfield of data mining that takes into consideration costs. The ultimate objective of cost-sensitive algorithms is to minimize the total cost. Note that cost needs to be understood in an abstract form, and it could be defined as anything that causes prejudice.

We find in the literature ten different costs that can be taken into consideration when we use Cost-sensitive algorithms [1]:

1. **Cost of Misclassification Errors**

   Cost of Misclassification happens when the model predicts incorrectly and assign a wrong class.

   a. Constant Error Cost

      It is the most researched error cost and happens with the error cost is constant.

   b. Conditional Error Cost

      Conditional Error Cost is a type the error that may be conditional on the circumstances.

      i. Error cost conditional on individual case
      ii. Error cost conditional of classification
      iii. Error cost conditional of classification of other cases
      iv. Error cost conditional of classification of other cases

2. **Cost of test**

Cost of test occurs when we need to decide whether is worthwhile to perform the test. If the misclassification error is greater than the cost, then it is convenient to perform the test. Otherwise, it is not convenient.

   a. Constant Test Cost

      Cost of asking the teacher is constant.

   b. Conditional teacher Cost

      The cost may increase change with circumstances of the case

3. **Cost of intervention**

   Cost of intervention means the effort required to manipulate a process.

   a. Constant intervention Cost

      Cost of intervention is constant

   b. Conditional intervention cost

4. **Cost of Unwanted Achievements**

   Unwanted Achievements are the unexpected consequences of performing the test. For instance, a model with 99% of accuracy and 1% of misclassification. 1% of the time will obtain Unwanted Achievements.

   a. Constant Unwanted Achievements Cost

      Cost of Unwanted Achievements is constant.

   b. Conditional intervention cost

5. **Cost of Computation**

   Computation is a limited resource and depending on its complexity the cost may differ

   a. Static Complexity
      i. Size complexity
      ii. Structural complexity
   b. Dynamic Complexity
      i. Time complexity
      ii. Space complexity
   c. Training Complexity
   d. Testing Complexity

6. **Cost of cases**

Cost of cases relates to the problem that comes with the availability of data and the small datasets. Also, the cost of acquiring more data. Static Complexity .

a. Cost of Cases for a Batch Learner

b. Cost of Cases for an Incremental

7. **Human-Computer Interaction Cost**

Cost of a human person using inductive learning software.

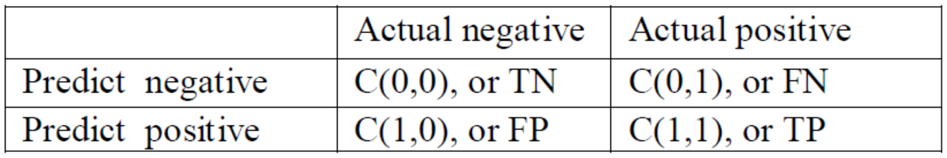
a. HCI Cost of Data Engineering
b. HCI Cost of Parameter Setting
c. HCI Cost of Analysis of Learned Models
d. HCI Cost of Incorporating Domain Knowledge

8. **Cost of Instability**

Stability is defined by two batches of data are generated from the same physical process. For a model to gain understanding of the underlying process that generated the data, it is essential that the model should be stable.

4.2 COST-SENSITIVE THEORY

Misclassification cost is an essential factor on cost-sensitive algorithms [10][11]. To illustrate how cost-sensitive learning works, it is assumed binary classification. In the cost matrix is denoted FP as false positive (predicted positive but it is negative), FN as false negative (predicted negative but it is positive), TP as true positive (predicted positive and it is possible) and TN as true negative (predicted negative and it is negative). It is also used the notation C(i,j) to show misclassification cost, where 1 stands for positive and 0 stands for negative. Note that misclassification cost values can be given by experts.



	Actual negative	Actual positive
Predict negative	C(0,0), or TN	C(0,1), or FN
Predict positive	C(1,0), or FP	C(1,1), or TP

Figure.1 Cost matrix for binary classification.

C(0,0) and C(1,1) or TN (True negative) and TP (true positive) is typically seen as good since the cases are classed correctly by the algorithm. On the other hand, C(1,0) and C(0,1) or FP (False positive) and FN (False negative) are generally seen as bad since these cases are classes incorrectly by the algorithm and those outcomes represent what is defined as cost.

The ultimate goal of the cost-sensitive algorithms is to minimize the costs of misclassification or in other words reduce C(1,0) and C(0,1) or FP and FN. This is the minimum expected cost principle $R(i \mid x)$.

$$R(i \mid x) = \sum_{j} P(j \mid x)\, C(i,j) \quad (1)$$

Where $P(j \mid x)$ is the probability estimation of misclassification. The classifier will class a case into positive if the addition of the FP and the TP is lower than the addition of TN and FN.

$$P(0|x)C(1,0) + P(1|x)C(1,1) \leq P(0|x)C(0,0) + P(1|x)C(0,1)$$

Or

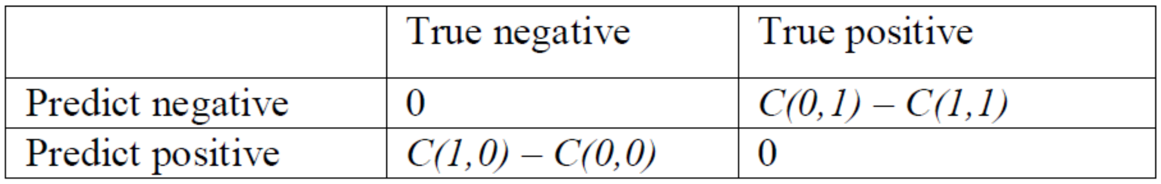
$$P(0|x)C(1,0) - C(0,0) \leq P(1|x)C(0,1) - C(1,1)$$

If a constant is added into the column of the cost matrix such as C (0,0) and C(1,1).

$$C\ (0,0) = C\ (1,1) = 0$$

A simple cost matrix comes up, where the classifier will class a case into positive if the FP C(1,0) is lower than the FN C(0,1).

$$P(0|x)C(1,0) \leq P(1|x)C(0,1) - C(0,1)$$



	True negative	True positive
Predict negative	0	*C(0,1) – C(1,1)*
Predict positive	*C(1,0) – C(0,0)*	0

Figure.2. Simplified cost matrix for binary classification.

As $P(0|x) = 1 - P\ (1|x)$ we can obtain a threshold P* for the classifier to class a case into positive if into positive if $P(1|x) \geq p$, where

$$P^* = \frac{\mathrm{C(1,0)}}{\mathrm{C(1,0)+C(0,1)}} = \frac{\mathrm{FP}}{\mathrm{FP+FN}} \ (2)$$

From this idea we can deduct that if a cost-insensitive classifier can produce a posterior probability estimation p(1|x), we can make it cost-sensitive by simply choosing the classification threshold according to (2), and classify any example to be positive whenever P(1|x) ≥ p*.

Note that Traditional cost-insensitive algorithms are built to class in terms of a default fixed threshold of 0.5[11]. In order to convert the algorithms into cost-sensitive we need to use a technique called rebalance which is to keep all the positives since they are considered as rare cases. On the side, negatives are multiplied by C(1,0)/C(0,1) = FP/FN.

If C(1,0)/C(0,1) = FP/FN is lower than 1 is called under-sampling. However, if C(1,0)/C(0,1) = FP/FN is 1 is called proportional sampling, where positive and negative cases are sampled by the ratio of: p(1) FN : p(0) FP (3) . Whereas if /FN is greater than 1 is called over-sampling. Most of the meta-learning approaches uses thresholding of (2) or (3).

4.3 COST-SENSITIVE METHODS

Cost-sensitive algorithms are divided into two categories direct and indirect or wrapper method or cost-sensitive meta-learning. The direct method consists of designing a Cost-sensitive classifier, whereas indirect or wrapper method consists of using a wrapper to convert a Cost-insensitive algorithm into a Cost-sensitive.

The direct method is utilized mainly to introduce the misclassification cost into the algorithm. The most relevant work on the field is the cost-sensitive decision trees [12] that builds the decision tree minimizing the expected total cost instead of the conventional use of the entropy to build the tree. There are also other works done on the direct method such as ICET [2].

The indirect or wrapper method or cost-sensitive meta-learning aims to convert a cost-insensitive algorithm into a cost-sensitive one without altering any of the parameters. That is being done by pre-processing the tanning data and post-processing the outcome. The indirect or wrapper method or cost-sensitive meta-learning has 3 main subfields, which are Boosting, Bagging and sampling.

The idea of boosting [13] is converting a set of weak learners into strong ones by training the weak learners. There are three types of boosting algorithms AdaBoost [14], AdaCost[15] and Weighting[16].

Bagging aims to produce an improved outcome with reduced variance and better accuracy by bootstrapping the samples in the training set. There are two types of Bagging methods MetaCost[17] and Costing[18].

Sampling aims to solve the problem of imbalance data by changing the number of rare cases in the training set based on the cost [19]. There are two sampling techniques which are random sampling and determinate sampling. Random sampling is to modify the distribution randomly, whereas the determinate sampling is to alter the distribution in a determined manner. Both techniques [20] can be used using Over-sampling, which increases the number of rare-classes and Under-sampling, which decreases the number of rare-classes.

5 RESEARCH METHODOLOGY

Several research methodologies have been used in the project:

On the introduction and the background, a **conceptual research methodology** was used to introduce the concepts and the most relevant literature in the field and to introduce the current state of art, and to introduce the cost-sensitive algorithms theory. On the exploration dataset was used a **descriptive research methodology** to describe the data, make sense of it, and show the importance of the attributes. The core of the project was made using the **quantitative research methodology** to deploy the cost-sensitive and insensitive algorithms on the online marketing dataset and compare them. Eventually, an **analytical research methodology** was used to evaluate the results obtained critically.

6 DATA EXPLORATION

6.1 DATA DESCRPTION

In this section, the dataset will be introduced, and it will be explored for further manipulation.

The data selected is a dataset in Online Marketing (Online Shoppers Purchasing Intention Dataset) [21]. The dataset used was found on https://archive.ics.uci.edu.

"The dataset consists of feature vectors belonging to 12,330 sessions. The dataset was formed so that each session would belong to a different user in 1 year to avoid any tendency to a specific campaign, special day, user profile, or period. " [22]

The dataset is made up of 12331 rows and 13 columns or attributes. The 18 attributes which 10 of them are numerical and 8 categorical attributes.

- Administrative
- Administrative_Duration
- Informational
- Informational_Duration
- ProductRelated
- ProductRelated_Duration
- BounceRates
- ExitRates
- PageValues
- SpecialDay
- Month
- OperatingSystems
- Browser
- Region

- TrafficType
- VisitorType
- Weekend
- Revenue

“Administrative", "Administrative Duration", "Informational", "Informational Duration", "Product Related" and "Product-Related Duration" represents the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories. The values of these features are derived from the URL information of the pages visited by the user and updated in real-time when a user takes action, e.g. moving from one page to another.

The "Bounce Rate", "Exit Rate" and "Page Value" features represent the metrics measured by "Google Analytics" for each page in the e-commerce site. The value of "Bounce Rate" feature for a web page refers to the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session.

The value of "Exit Rate" feature for a specific web page is calculated as for all pageviews to the page, the percentage that was the last in the session.

The "Page Value" feature represents the average value for a web page that a user visited before completing an e-commerce transaction

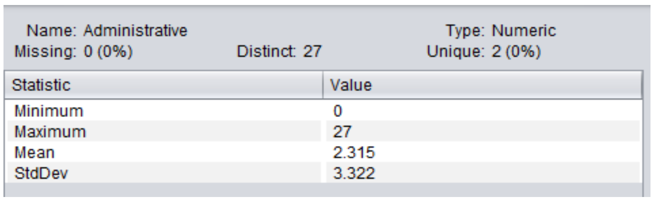
The "Special Day" feature indicates the closeness of the site visiting time to a specific special day (e.g. Mother’s Day, Valentine's Day) in which the sessions are more likely to be finalized with the transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentina’s day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8.

The dataset also includes the operating system, browser, region, traffic type, visitor type as returning or new visitor, a Boolean value indicating whether the date of the visit is weekend, and month of the year.” [22]

Finally, the Revenue attribute is whether the customer purchase or not. This attribute will be used as the class label.

6.2 ATTRIBUTE EXPLORATION

- Administrative



Name: Administrative
Missing: 0 (0%)
Distinct: 27
Type: Numeric
Unique: 2 (0%)

Statistic	Value
Minimum	0
Maximum	27
Mean	2.315
StdDev	3.322

Figure.3. Administrative attribute statistics. Source: Weka.

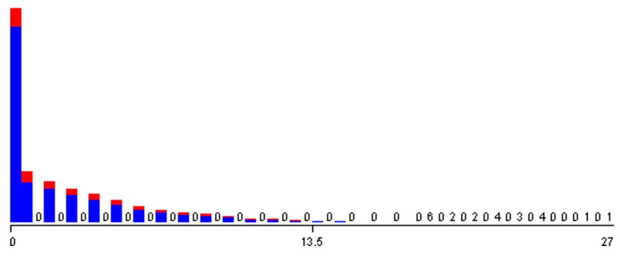
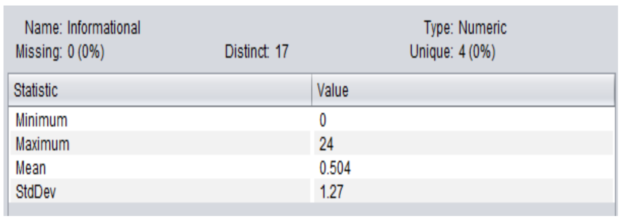


Figure.4. Administrative histogram. Source: Weka.

The Administrative attribute fluctuates in a range of 0 and 27 with an average of 2.3 and a standard deviation of 3.2. As we can see in figure 4, most of the sales (red part of the histogram) are made under 7 pages visited. The Administrative attribute is a a numerical attribute.

- Informational



Name: Informational | Type: Numeric
Missing: 0 (0%) | Distinct: 17 | Unique: 4 (0%)

Statistic	Value
Minimum	0
Maximum	24
Mean	0.504
StdDev	1.27

Figure.5. Informational attribute statistics. Source: Weka.

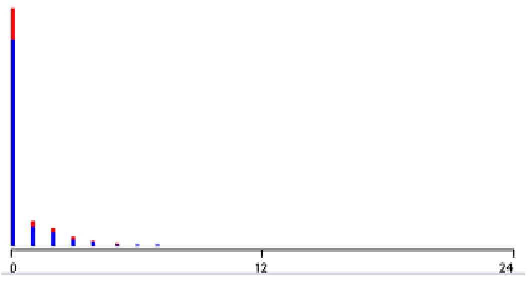
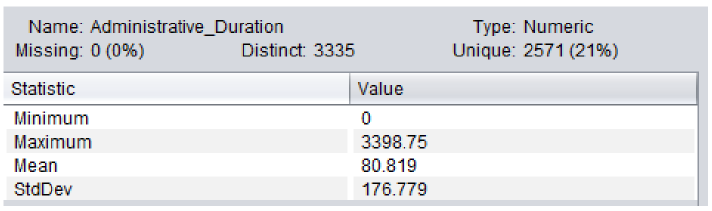


Figure.6. Informational histogram. Source: Weka.

The Informational attribute vary in a range of 0 and 24 with an average of 0.5 and a standard deviation of 1.2. As we can see in figure 6, most of the sales (red part of the histogram) are made under 6 pages visited. The Informational attribute is a numerical attribute.

- Administrative Duration



Name: Administrative_Duration | Type: Numeric
Missing: 0 (0%) | Distinct: 3335 | Unique: 2571 (21%)

Statistic	Value
Minimum	0
Maximum	3398.75
Mean	80.819
StdDev	176.779

Figure.7. Administrative duration attribute statistics. Source: Weka.

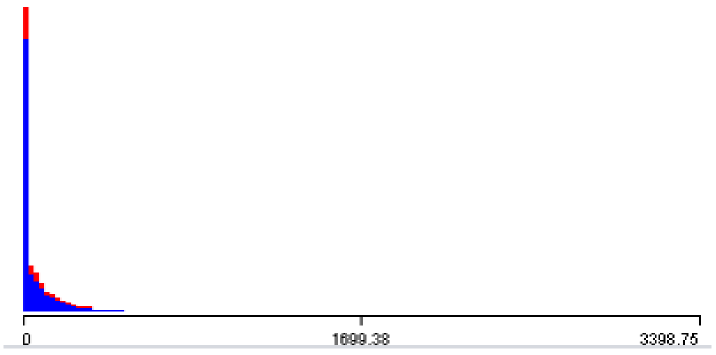
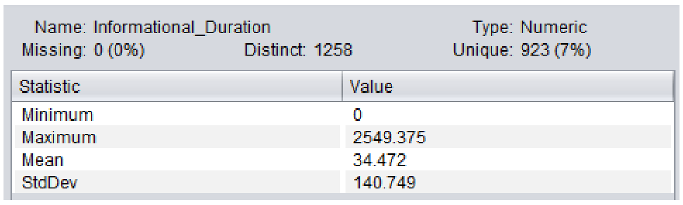


Figure.8. Administrative duration histogram. Source: Weka.

The Administrative duration attribute shift in a range of 0 and 3398 seconds with a mean of 80 seconds and a standard deviation of 176 seconds. The Administrative duration attribute is a numerical attribute.

- Informational Duration



Name: Informational_Duration
Type: Numeric
Missing: 0 (0%)
Distinct: 1258
Unique: 923 (7%)

Statistic	Value
Minimum	0
Maximum	2549.375
Mean	34.472
StdDev	140.749

Figure.9. Informational duration attribute statistics. Source: Weka.

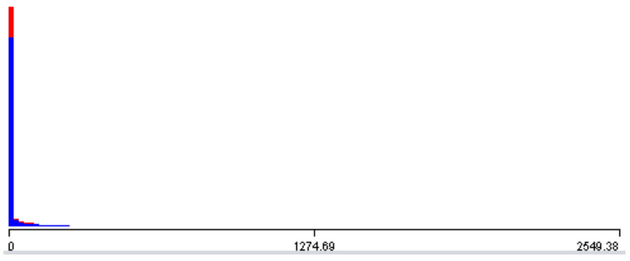
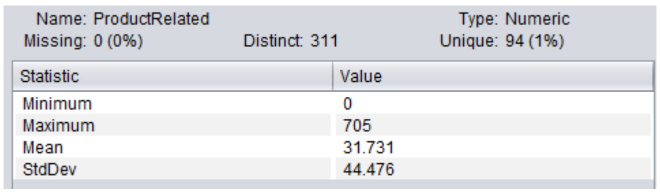


Figure.10. Informational duration histogram. Source: Weka.

The Informational duration attribute changes in a range of 0 and 2549 seconds with a mean of 34 seconds and a standard deviation of 140 seconds. The Informational duration attribute is a numerical attribute.

- Product Related


Name: ProductRelated Type: Numeric
Missing: 0 (0%) Distinct: 311 Unique: 94 (1%)

Statistic	Value
Minimum	0
Maximum	705
Mean	31.731
StdDev	44.476

Figure.11. Product Related attribute statistics. Source: Weka.

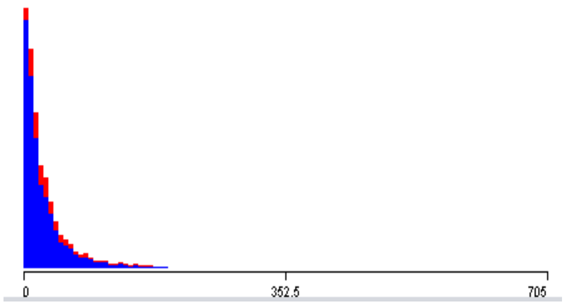
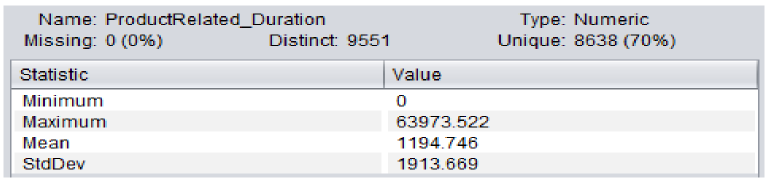


Figure.12. Product Related histogram. Source: Weka.

The Administrative duration attribute fluctuates in a range of 0 and 3398 with a mean of 80 and a standard deviation of 176. The Administrative duration attribute is a numerical attribute.

- Product Related Duration


Name: ProductRelated_Duration Type: Numeric
Missing: 0 (0%) Distinct: 9551 Unique: 8638 (70%)

Statistic	Value
Minimum	0
Maximum	63973.522
Mean	1194.746
StdDev	1913.669

Figure.13. Product related duration attribute statistics. Source: Weka.

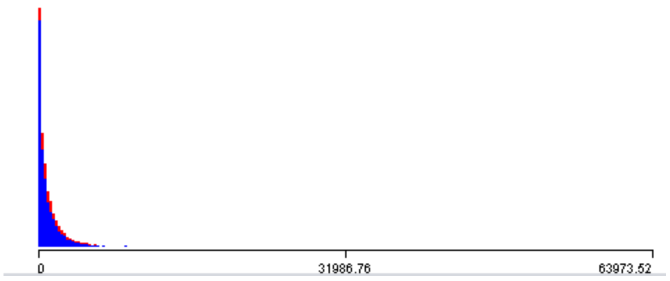
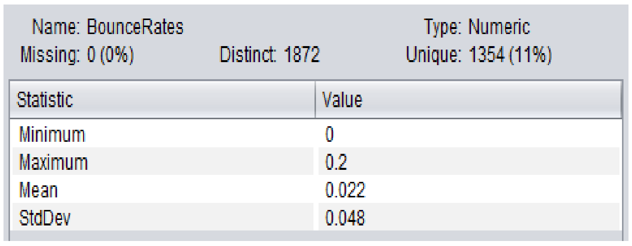


Figure.14. Product related duration histogram. Source: Weka.

The product related duration attribute shift in a range of 0 and 63973 seconds with a mean of 1194 seconds and a standard deviation of 1913 seconds. The product related duration attribute is a numerical attribute.

- Bounce Rates



Name: BounceRates
Missing: 0 (0%)
Distinct: 1872
Type: Numeric
Unique: 1354 (11%)

Statistic	Value
Minimum	0
Maximum	0.2
Mean	0.022
StdDev	0.048

Figure.14. Bounce Rates attribute statistics. Source: Weka.

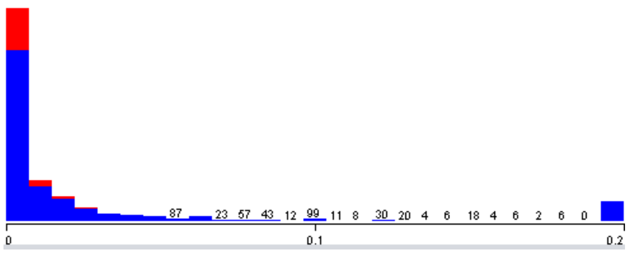
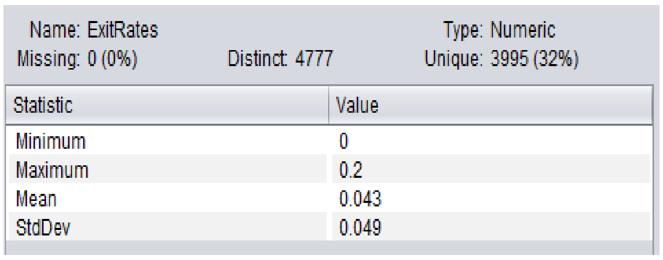


Figure.15. Bounce Rates histogram. Source: Weka.

The Bounce rate attribute changes in a range of 0 and 0.2 with an average of 0.02 and a standard deviation of 0.04. The Bounce rate attribute is a numerical attribute.

- Exit Rates



Name: ExitRates
Missing: 0 (0%)
Distinct: 4777
Type: Numeric
Unique: 3995 (32%)

Statistic	Value
Minimum	0
Maximum	0.2
Mean	0.043
StdDev	0.049

Figure.16. Exit rates attribute statistics. Source: Weka.

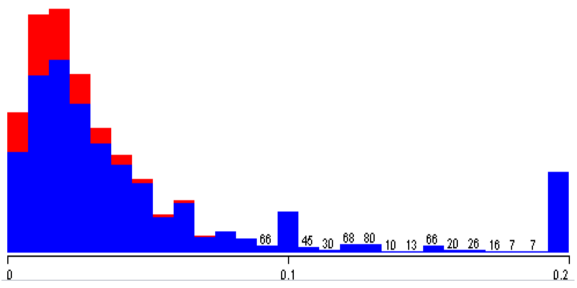
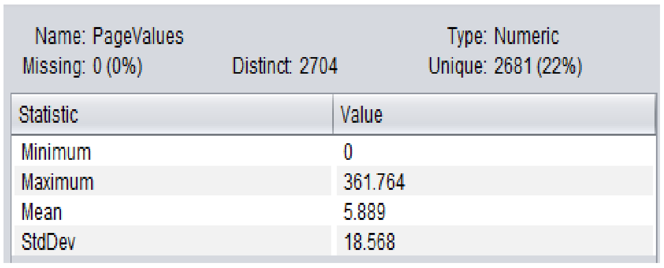


Figure.17. Exit rates histogram. Source: Weka.

The Exit rate attribute changes in a range of 0 and 0.2 with an average of 0.04 and a standard deviation of 0.04. The Exit rate attribute is a numerical attribute

- Page Values



Name: PageValues
Missing: 0 (0%)
Distinct: 2704
Type: Numeric
Unique: 2681 (22%)

Statistic	Value
Minimum	0
Maximum	361.764
Mean	5.889
StdDev	18.568

Figure.18. Administrative attribute statistics. Source: Weka.

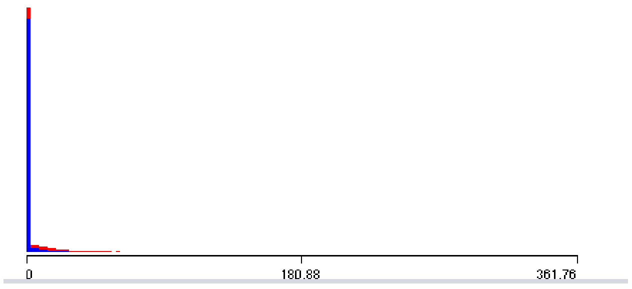
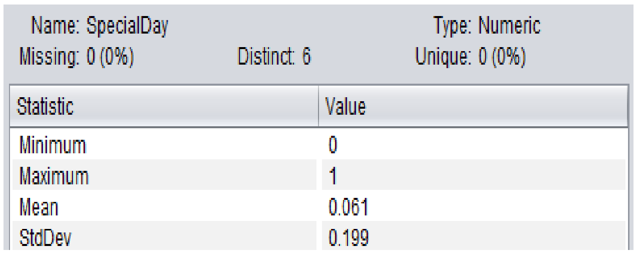


Figure.19. Administrative histogram. Source: Weka.

The Page values attribute vary in a range of 0 and 361 with an average of 5.8 and a standard deviation of 18.5. The Page values attribute is a numerical attribute.

- Special Day


Name: SpecialDay | Type: Numeric
Missing: 0 (0%) | Distinct: 6 | Unique: 0 (0%)

Statistic	Value
Minimum	0
Maximum	1
Mean	0.061
StdDev	0.199

Figure.20. special day attribute statistics. Source: Weka.

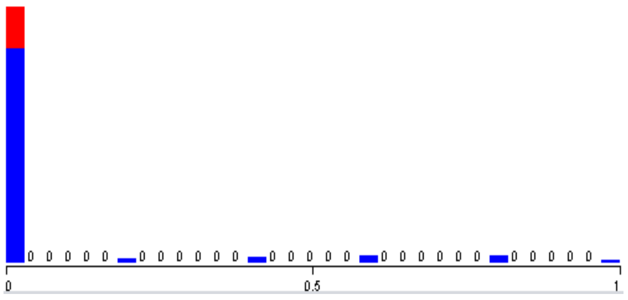
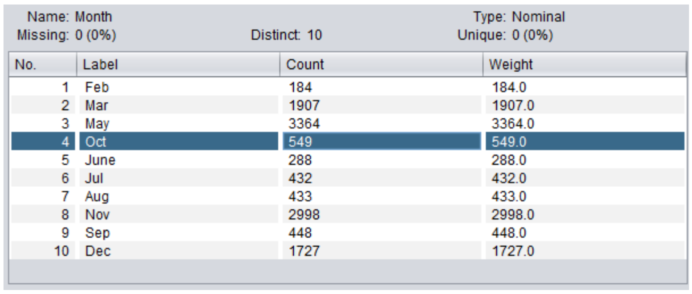


Figure.21. Administrative histogram. Source: Weka.

The Special day attribute changes in a range of 0 and It is a binary attribute. The Special day attribute is a numerical attribute.

- Month


Name: Month | Type: Nominal
Missing: 0 (0%) | Distinct: 10 | Unique: 0 (0%)

No.	Label	Count	Weight
1	Feb	184	184.0
2	Mar	1907	1907.0
3	May	3364	3364.0
4	Oct	549	549.0
5	June	288	288.0
6	Jul	432	432.0
7	Aug	433	433.0
8	Nov	2998	2998.0
9	Sep	448	448.0
10	Dec	1727	1727.0

Figure.22. Month attribute statistics. Source: Weka.

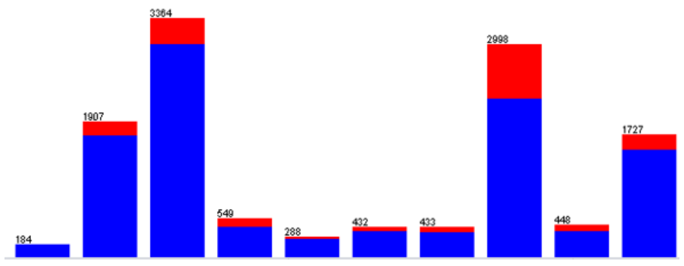
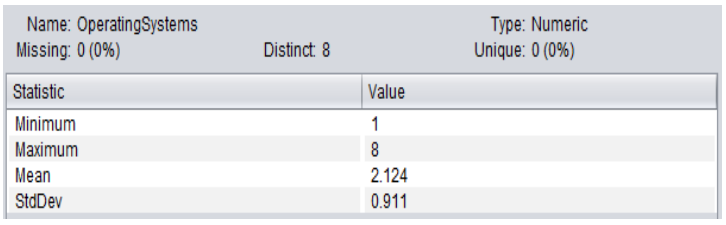


Figure.23. Administrative histogram. Source: Weka.

Figure.23. Month histogram. Source: Weka.

The month shows the sales by month. The month attribute is a nominal attribute.

- Operating Systems



Name: OperatingSystems		Type: Numeric
Missing: 0 (0%)	Distinct: 8	Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	8
Mean	2.124
StdDev	0.911

Figure.24. Operation Systems attribute statistics. Source: Weka.

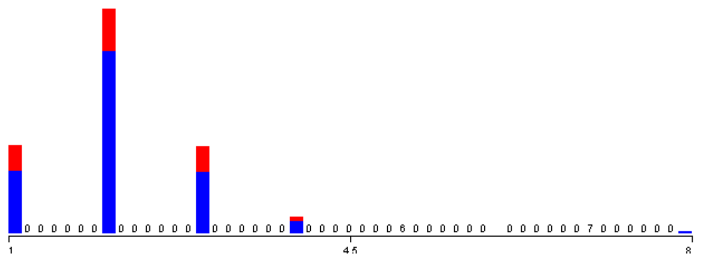
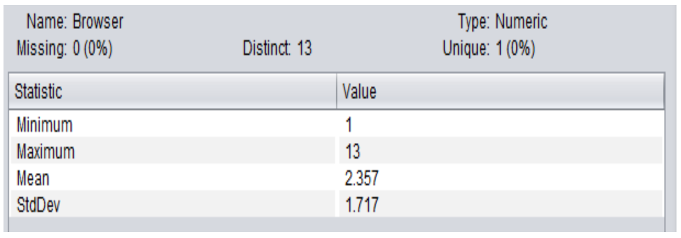


Figure.25. Operation Systems histogram. Source: Weka.

The Operating system illustrates the sales by sort of Operating system. The Operating system attribute is a nominal attribute.

- Browser



Name: Browser		Type: Numeric
Missing: 0 (0%)	Distinct: 13	Unique: 1 (0%)

Statistic	Value
Minimum	1
Maximum	13
Mean	2.357
StdDev	1.717

Figure.26. Browser statistics. Source: Weka.

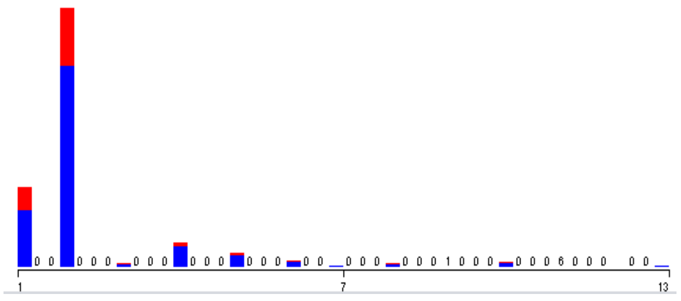
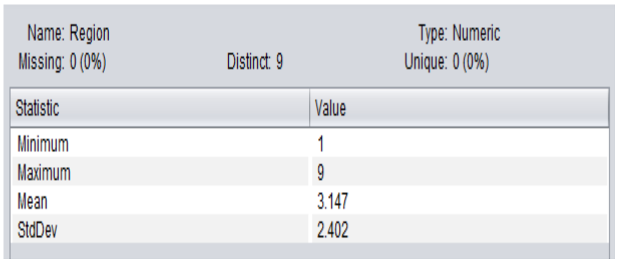


Figure.27. Browser histogram. Source: Weka.

The Browser illustrates the sales by sort of Browser. The Browser attribute is a nominal attribute.

- Region



Name: Region
Type: Numeric
Missing: 0 (0%)
Distinct: 9
Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	9
Mean	3.147
StdDev	2.402

Figure.28. Region attribute statistics. Source: Weka.

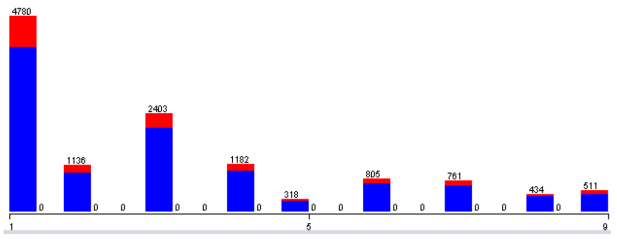
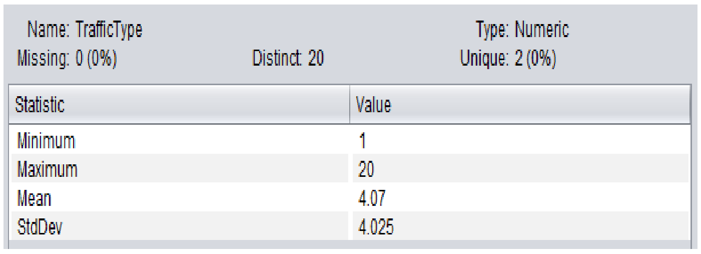


Figure.29. Region histogram. Source: Weka.

The Region illustrates the sales by the different Regions. The Region attribute is a nominal attribute.

- Traffic Type



Name: TrafficType | Type: Numeric
Missing: 0 (0%) | Distinct: 20 | Unique: 2 (0%)

Statistic	Value
Minimum	1
Maximum	20
Mean	4.07
StdDev	4.025

Figure.30. Traffic type statistics. Source: Weka.

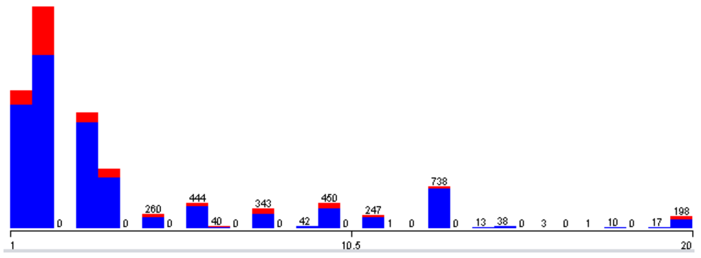
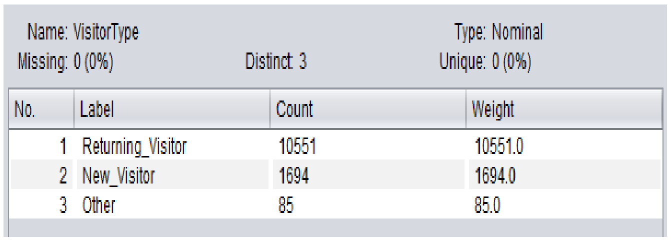


Figure.31. Traffic type histogram. Source: Weka.

The traffic type illustrates the sales by the different traffic type. The traffic type attribute is a nominal attribute.

- Visitor Type



Name: VisitorType | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

No.	Label	Count	Weight
1	Returning_Visitor	10551	10551.0
2	New_Visitor	1694	1694.0
3	Other	85	85.0

Figure.32. Visitor type attribute statistics. Source: Weka.

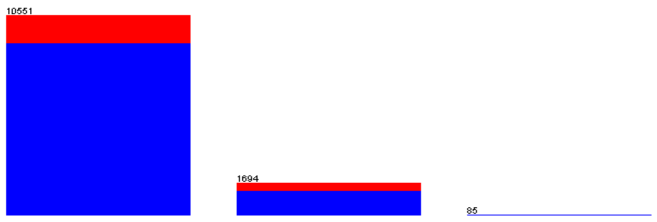
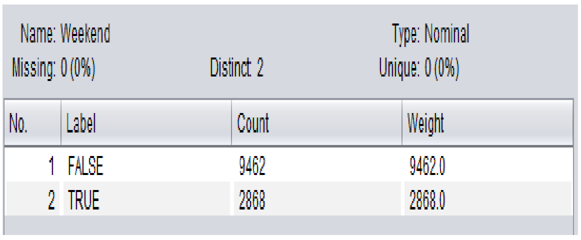


Figure.33. Visitor type histogram. Source: Weka.

The Visitor type illustrates the sales by the different Visitor type. The Visitor type attribute is a nominal attribute.

- Weekend



Name: Weekend
Missing: 0 (0%) Distinct: 2 Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	FALSE	9462	9462.0
2	TRUE	2868	2868.0

Figure.34. Weekend attribute statistics. Source: Weka.

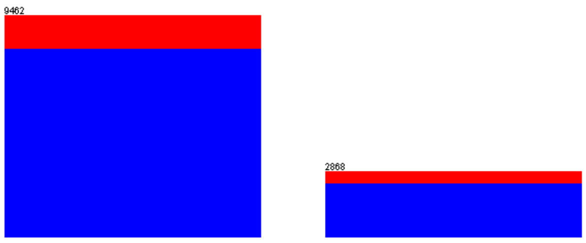
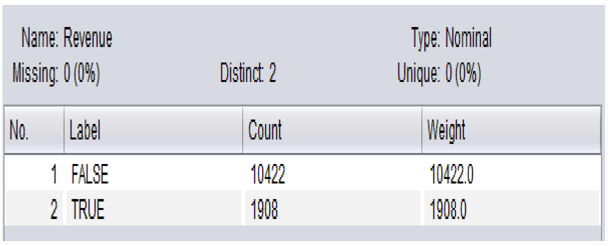


Figure.35. Weekend histogram. Source: Weka.

The Weekend illustrates the sales by either Weekend or not Weekend. The Weekend attribute is a nominal attribute.

- Revenue



Name: Revenue
Missing: 0 (0%) Distinct: 2 Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	FALSE	10422	10422.0
2	TRUE	1908	1908.0

Figure.36. Revenue attribute statistics. Source: Weka.

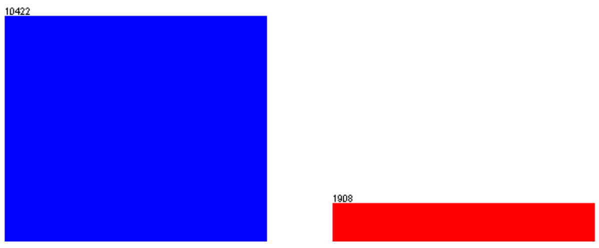


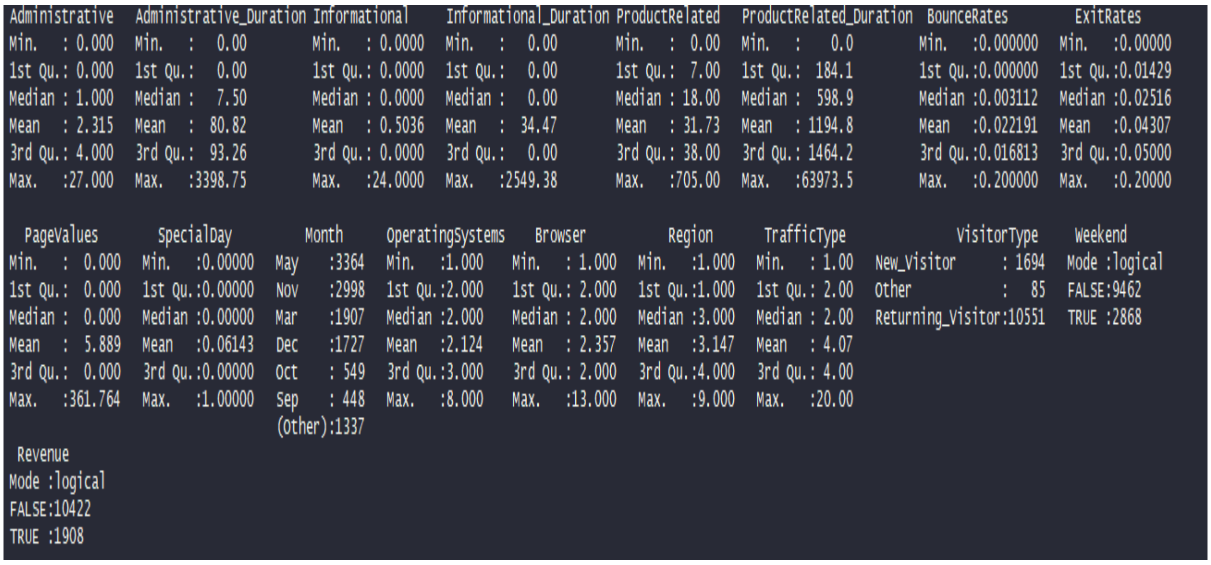
Figure.37. Revenue histogram. Source: Weka.

The revenue illustrates whether the user purchased or not. Blue means that user did not purchase, whereas the red colour means the user purchased. The revenue attribute is a nominal attribute.

7 DATA PREPROCESSING

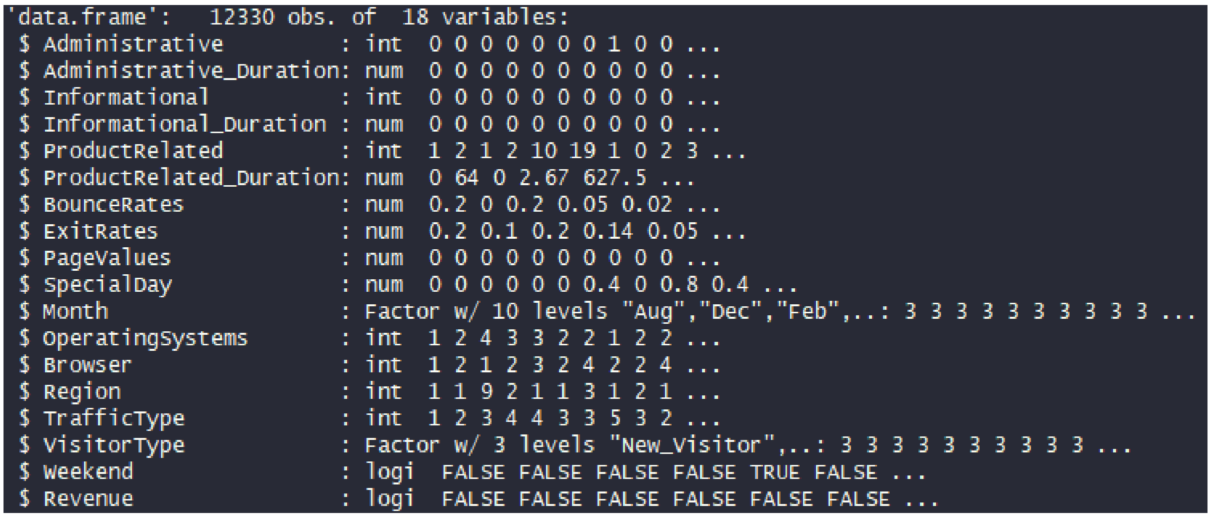
This section will show how the data has been pre-processed to prepare it for using in the algorithms later. Please note that the pre-processing has been performed using the R programming language for the convenience of the author of this project.

An initial inspection on the dataset was needed to see whether there were missing values or faulty data.



```
Administrative   Administrative_Duration Informational    Informational_Duration ProductRelated   ProductRelated_Duration  BounceRates        ExitRates
Min.   : 0.000   Min.   :   0.00         Min.   : 0.0000  Min.   :   0.00        Min.   :  0.00   Min.   :    0.0         Min.   :0.000000   Min.   :0.00000
1st Qu.: 0.000   1st Qu.:   0.00         1st Qu.: 0.0000  1st Qu.:   0.00        1st Qu.:  7.00   1st Qu.:  184.1         1st Qu.:0.000000   1st Qu.:0.01429
Median : 1.000   Median :   7.50         Median : 0.0000  Median :   0.00        Median : 18.00   Median :  598.9         Median :0.003112   Median :0.02516
Mean   : 2.315   Mean   :  80.82         Mean   : 0.5036  Mean   :  34.47        Mean   : 31.73   Mean   : 1194.8         Mean   :0.022191   Mean   :0.04307
3rd Qu.: 4.000   3rd Qu.:  93.26         3rd Qu.: 0.0000  3rd Qu.:   0.00        3rd Qu.: 38.00   3rd Qu.: 1464.2         3rd Qu.:0.016813   3rd Qu.:0.05000
Max.   :27.000   Max.   :3398.75         Max.   :24.0000  Max.   :2549.38        Max.   :705.00   Max.   :63973.5         Max.   :0.200000   Max.   :0.20000

  PageValues        SpecialDay           Month      OperatingSystems    Browser           Region        TrafficType              VisitorType      Weekend
Min.   :  0.000   Min.   :0.00000   May    :3364   Min.   :1.000   Min.   : 1.000   Min.   :1.000   Min.   : 1.00   New_Visitor      : 1694   Mode :logical
1st Qu.:  0.000   1st Qu.:0.00000   Nov    :2998   1st Qu.:2.000   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 2.00   Other            :   85   FALSE:9462
Median :  0.000   Median :0.00000   Mar    :1907   Median :2.000   Median : 2.000   Median :3.000   Median : 2.00   Returning_Visitor:10551   TRUE :2868
Mean   :  5.889   Mean   :0.06143   Dec    :1727   Mean   :2.124   Mean   : 2.357   Mean   :3.147   Mean   : 4.07
3rd Qu.:  0.000   3rd Qu.:0.00000   Oct    : 549   3rd Qu.:3.000   3rd Qu.: 2.000   3rd Qu.:4.000   3rd Qu.: 4.00
Max.   :361.764   Max.   :1.00000   Sep    : 448   Max.   :8.000   Max.   :13.000   Max.   :9.000   Max.   :20.00
                                    (Other):1337
 Revenue
Mode :logical
FALSE:10422
TRUE :1908
```

Figure.37. Dataset inspection on the missing values and faulty data. Source: RStudio.



```
'data.frame':	12330 obs. of  18 variables:
 $ Administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
 $ Administrative_Duration: num  0 0 0 0 0 0 0 0 0 0 ...
 $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Informational_Duration : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ProductRelated         : int  1 2 1 2 10 19 1 0 2 3 ...
 $ ProductRelated_Duration: num  0 64 0 2.67 627.5 ...
 $ BounceRates            : num  0.2 0 0.2 0.05 0.02 ...
 $ ExitRates              : num  0.2 0.1 0.2 0.14 0.05 ...
 $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SpecialDay             : num  0 0 0 0 0 0 0.4 0 0.8 0.4 ...
 $ Month                  : Factor w/ 10 levels "Aug","Dec","Feb",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ OperatingSystems       : int  1 2 4 3 3 2 2 1 2 2 ...
 $ Browser                : int  1 2 1 2 3 2 4 2 2 4 ...
 $ Region                 : int  1 1 9 2 1 1 3 1 2 1 ...
 $ TrafficType            : int  1 2 3 4 4 3 3 5 3 2 ...
 $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

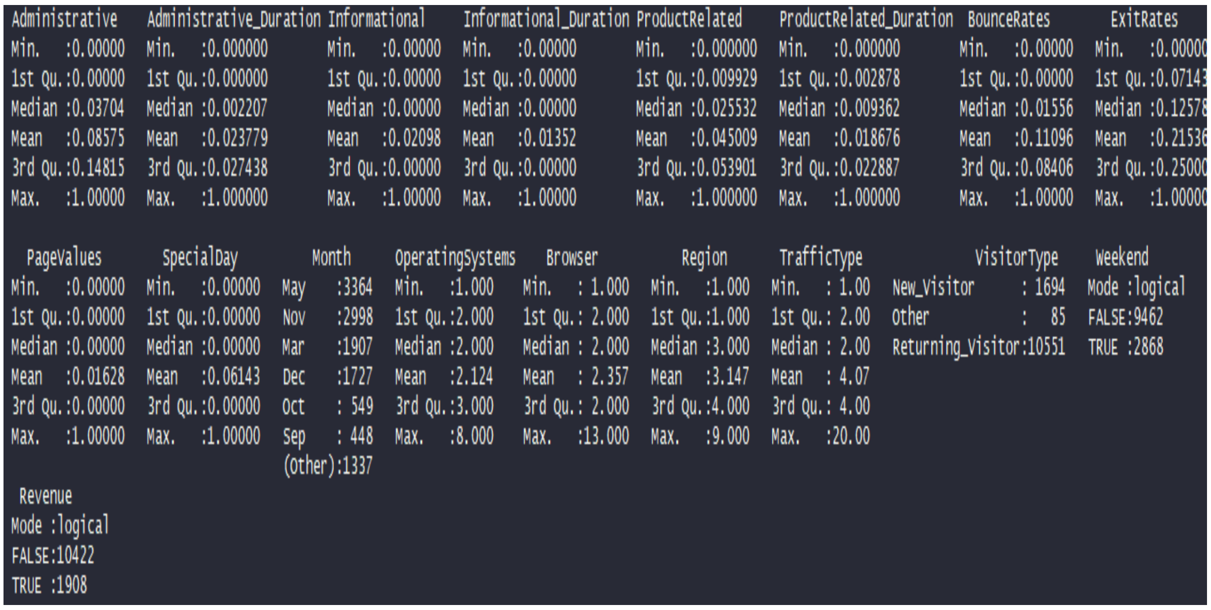
Figure.38. Dataset inspection on the category of the attributes. Source: RStudio.

We find that the dataset was already cleaned when It was downloaded from https://archive.ics.uci.edu, since there are no missing values, however in order to have the data ready to use we need to set the right category for the variables and we need to normalize some of the values of the attribute, otherwise they will distort by skewing the results.

First, we normalize [23] the values of the numerical attributes such as Administrative, Administrative_Duration, Informational, Informational_Duration, ProductRelated, ProductRelated_Duration, BounceRates, ExitRates, PageValues to adjust the values of the attributes to a common scale. The normalization allows to have all the values within a same range and to be able to compare them. It helps the algorithm to weight better all the factors and prevent values of distorting the results.

The type of normalization used for its simplicity is min-max normalization:

$x' = \frac{\mathrm{x-Min(x)}}{Max(x)-Min(x)}$ where Min-Max is [0,1]

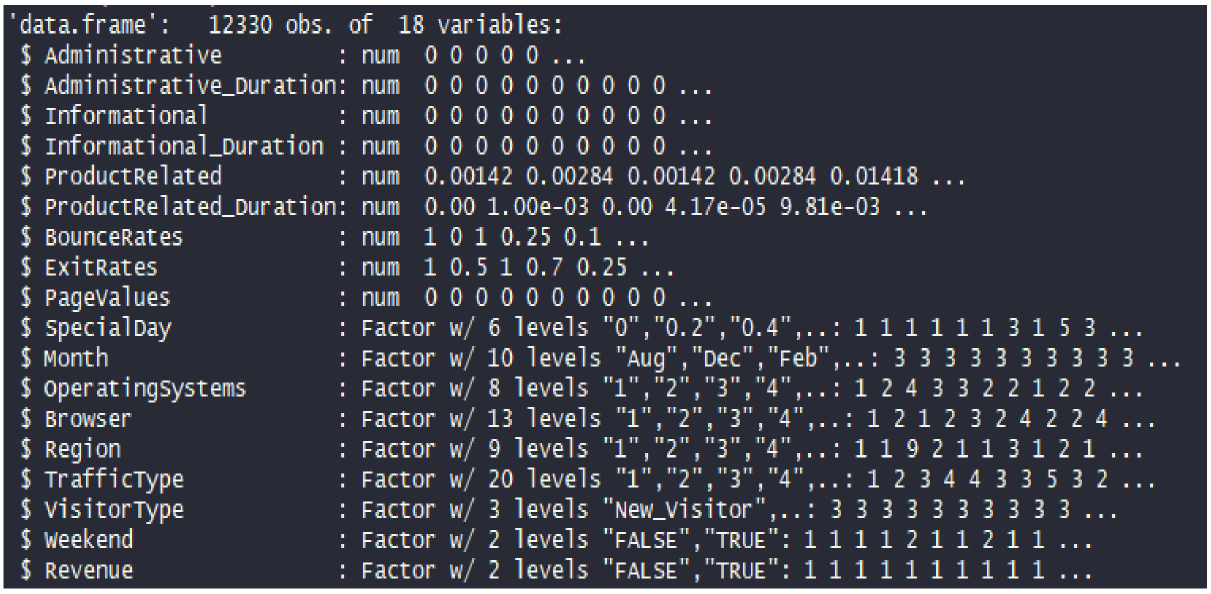


```
Administrative    Administrative_Duration Informational    Informational_Duration ProductRelated     ProductRelated_Duration  BounceRates       ExitRates
Min.   :0.00000   Min.   :0.000000        Min.   :0.00000  Min.   :0.00000        Min.   :0.000000   Min.   :0.000000         Min.   :0.00000   Min.   :0.00000
1st Qu.:0.00000   1st Qu.:0.000000        1st Qu.:0.00000  1st Qu.:0.00000        1st Qu.:0.009929   1st Qu.:0.002878         1st Qu.:0.00000   1st Qu.:0.07143
Median :0.03704   Median :0.002207        Median :0.00000  Median :0.00000        Median :0.025532   Median :0.009362         Median :0.01556   Median :0.12578
Mean   :0.08575   Mean   :0.023779        Mean   :0.02098  Mean   :0.01352        Mean   :0.045009   Mean   :0.018676         Mean   :0.11096   Mean   :0.21536
3rd Qu.:0.14815   3rd Qu.:0.027438        3rd Qu.:0.00000  3rd Qu.:0.00000        3rd Qu.:0.053901   3rd Qu.:0.022887         3rd Qu.:0.08406   3rd Qu.:0.25000
Max.   :1.00000   Max.   :1.000000        Max.   :1.00000  Max.   :1.00000        Max.   :1.000000   Max.   :1.000000         Max.   :1.00000   Max.   :1.00000

  PageValues        SpecialDay           Month       OperatingSystems    Browser           Region         TrafficType             VisitorType        Weekend
Min.   :0.00000   Min.   :0.00000   May      :3364   Min.   :1.000    Min.   : 1.000   Min.   :1.000   Min.   : 1.00   New_Visitor      : 1694   Mode :logical
1st Qu.:0.00000   1st Qu.:0.00000   Nov      :2998   1st Qu.:2.000    1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 2.00   Other            :   85   FALSE:9462
Median :0.00000   Median :0.00000   Mar      :1907   Median :2.000    Median : 2.000   Median :3.000   Median : 2.00   Returning_Visitor:10551   TRUE :2868
Mean   :0.01628   Mean   :0.06143   Dec      :1727   Mean   :2.124    Mean   : 2.357   Mean   :3.147   Mean   : 4.07
3rd Qu.:0.00000   3rd Qu.:0.00000   Oct      : 549   3rd Qu.:3.000    3rd Qu.: 2.000   3rd Qu.:4.000   3rd Qu.: 4.00
Max.   :1.00000   Max.   :1.00000   Sep      : 448   Max.   :8.000    Max.   :13.000   Max.   :9.000   Max.   :20.00
                                    (Other):1337
 Revenue
Mode :logical
FALSE:10422
TRUE :1908
```

Figure.39. Dataset with values normalized using the Min-Max method. Source: RStudio.

On the other hand, in order to run the algorithm correctly, we need to set the correct category for the attributes.

As we could see in figure 38, the categories are numeric, integer, factor and logical. However, we need them to be either numerical for numeric attributes or factor for categorical attributes. It is essential for the algorithm to differentiate the numerical values from the factor and to be able to run the algorithm correctly without distortion.



```
'data.frame':   12330 obs. of  18 variables:
 $ Administrative         : num  0 0 0 0 0 ...
 $ Administrative_Duration: num  0 0 0 0 0 0 0 0 0 0 ...
 $ Informational          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Informational_Duration : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ProductRelated         : num  0.00142 0.00284 0.00142 0.00284 0.01418 ...
 $ ProductRelated_Duration: num  0.00 1.00e-03 0.00 4.17e-05 9.81e-03 ...
 $ BounceRates            : num  1 0 1 0.25 0.1 ...
 $ ExitRates              : num  1 0.5 1 0.7 0.25 ...
 $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SpecialDay             : Factor w/ 6 levels "0","0.2","0.4",..: 1 1 1 1 1 1 3 1 5 3 ...
 $ Month                  : Factor w/ 10 levels "Aug","Dec","Feb",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 1 2 2 ...
 $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 2 4 ...
 $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 1 2 1 ...
 $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 5 3 2 ...
 $ VisitorType            : Factor w/ 3 levels "New_Visitor",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Weekend                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 2 1 1 2 1 1 ...
 $ Revenue                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
```

Figure.40. Dataset with attributes with the correct category assigned and values normalized. Source: RStudio.

Finally, the pre-preceding is complete as the data is cleaned, the values are normalized, and the attributes are correctly classed.   The dataset is prepared to be used and extract information.

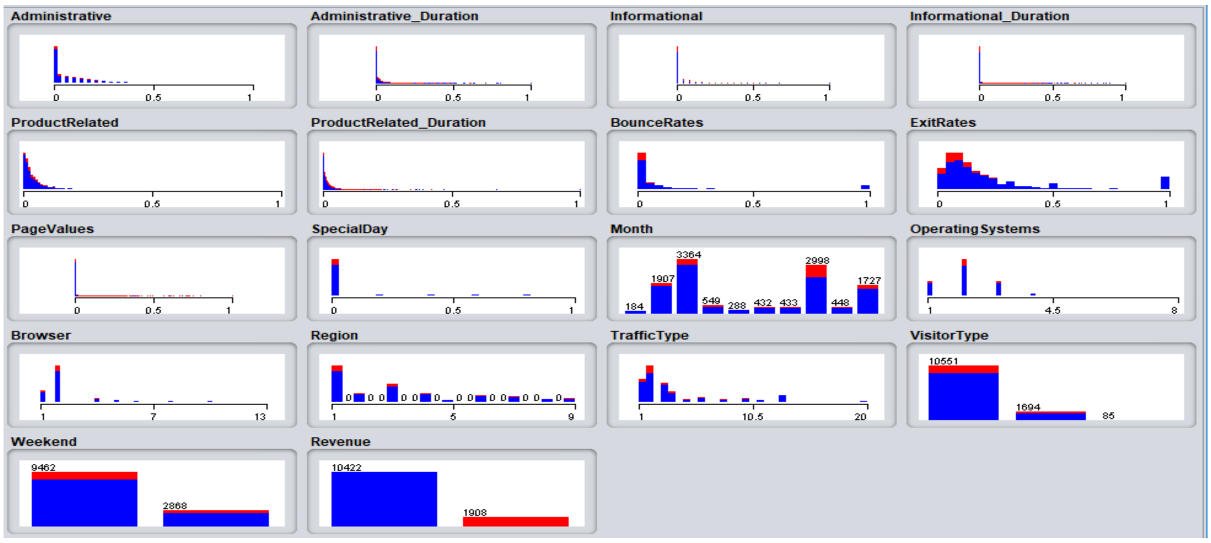


Figure.41. Overview of all the attributes after pre-processing. Source: Weka

8 APPLYING COST-SENSITIVE ALGORITHMS IN ONLINE MARKETING

This section aims to test the different cost-sensitive algorithms reviewed in the background section, such as the direct method and indirect or wrapper method using the Weka software.

Section 8.1 will build a direct method cost-sensitive model using a random forest algorithm. Sections 8.2 and 8.3 will develop indirect methods such as the boosting method using the addboostM1 classifier and the bagging method using decision trees. In section 8.4 will sample the dataset using the over-sampling technique to balance the data, and we will build a Random Forest. Besides, we will compare the results with a Random Forest with unbalanced data.

Section 8.5, we will evaluate the models regarding the cost, the accuracy and its online marketing goals.

The metrics used to evaluate the models under the cost point of view are as follows:

Correctly Classified Instances (Accuracy) = (TP + TN)/(TP+TN+FP+FN) → How good the model predicted the results

Incorrectly Classified Instances (Misclassification rate) = 1- Accuracy → Percentage of error in the model

Total cost= FP*weight + FN* weight → Total cost suffered because of misclassification

False positive rate = FP / (TP + TN) → Percentage of positive incorrectly classed

False negative rate = FN / (FN + TP) → Percentage of positive incorrectly classed

Roc Area = sensitivity/specificity → represents a sensitivity/specificity pair corresponding to a particular decision threshold.

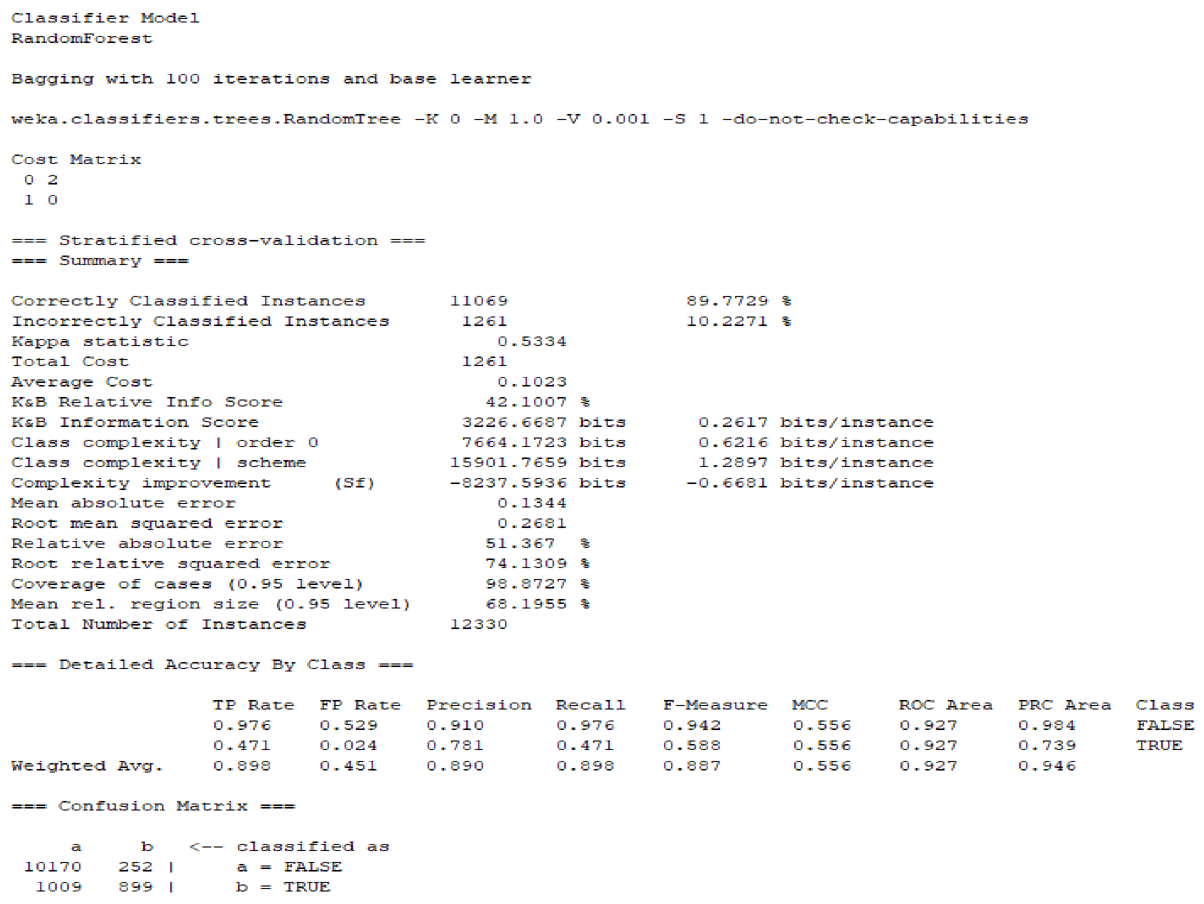
Note that all the models have been built using a black-box approach.

8.1 DIRECT METHOD - RANDOM FOREST

The direct method introduces misclassification cost into the algorithm and allows insensitive algorithms such as the Random Forest [24] to become cost sensitive.

The cost-sensitive Radom forest has been built using 100 decision trees to avoid overfitting and to reduce variance in the results. Also, the dataset was split in 66% training set, and 33% test set and a cross-validation 10-fold were used to train the model. The attribute used as a class label is revenue.

The first cost-sensitive Radom forest was build using a cost matrix, which penalized the false positive (number used is 2, and it is arbitrary, and it varies according to cost suffered) or customer that did not buy the product, but the model classed them as if they bought the product.



```
Classifier Model
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

Cost Matrix
 0 2
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11069               89.7729 %
Incorrectly Classified Instances      1261               10.2271 %
Kappa statistic                          0.5334
Total Cost                            1261
Average Cost                             0.1023
K&B Relative Info Score                 42.1007 %
K&B Information Score                 3226.6687 bits      0.2617 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            15901.7659 bits      1.2897 bits/instance
Complexity improvement     (Sf)      -8237.5936 bits     -0.6681 bits/instance
Mean absolute error                      0.1344
Root mean squared error                  0.2681
Relative absolute error                 51.367  %
Root relative squared error             74.1309 %
Coverage of cases (0.95 level)          98.8727 %
Mean rel. region size (0.95 level)      68.1955 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.976    0.529    0.910      0.976    0.942      0.556    0.927     0.984     FALSE
                 0.471    0.024    0.781      0.471    0.588      0.556    0.927     0.739     TRUE
Weighted Avg.    0.898    0.451    0.890      0.898    0.887      0.556    0.927     0.946

=== Confusion Matrix ===

     a     b   <-- classified as
 10170   252 |     a = FALSE
  1009   899 |     b = TRUE
```

Figure.41. Results of the Random Forest penalizing false positive. Source: Weka.

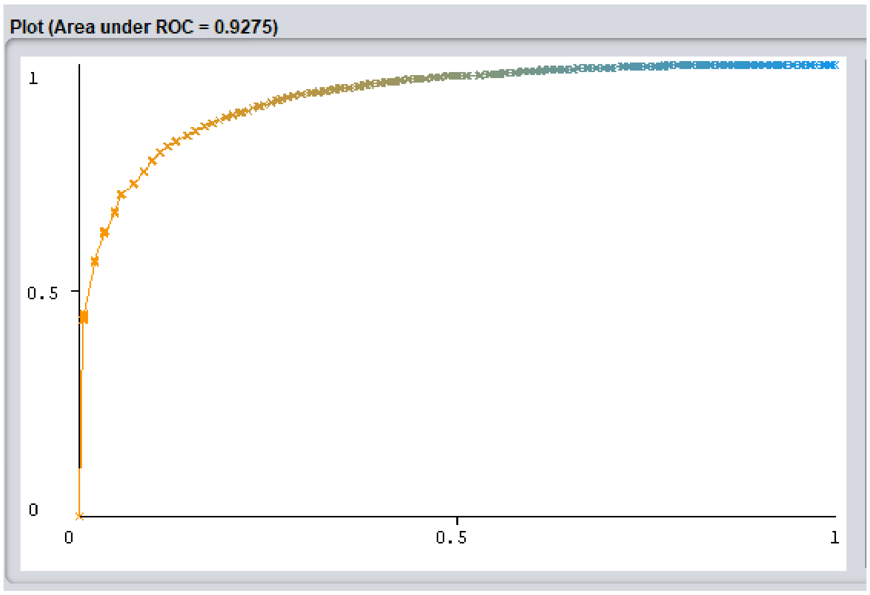


Figure.42. Roc plot of the Random Forest penalizing false positive. Source: Weka.

Correctly Classified Instances (Accuracy) = 89.77%

Incorrectly Classified Instances (Misclassification rate) =10.22%

Roc = 0.9275

Total cost= 1261

False positive rate = 0.529

False negative rate = 0.024

False positive instances=1009

False negative instances=252

We observe that the model correctly classified around 89.77% of the instances, whereas the misclassification was 10.22 % and with a false positive rate of 0.529 and a false negative rate of 0.024. The total cost was 1261, which there were 1009 False positive instances and 252 False negative instances, respectively. The model achieved a Roc of 0.9275.

The second cost-sensitive Radom forest built has penalized the false negative or the customer who bought the product, but the model classes them as if they did not buy.

```
Classifier Model
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

Cost Matrix
 0 1
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11114               90.1379 %
Incorrectly Classified Instances      1216                9.8621 %
Kappa statistic                          0.6194
Total Cost                            1216
Average Cost                             0.0986
K&B Relative Info Score                 32.9701 %
K&B Information Score                 2526.8869 bits      0.2049 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            11608.5301 bits      0.9415 bits/instance
Complexity improvement     (Sf)      -3944.3578 bits     -0.3199 bits/instance
Mean absolute error                      0.148
Root mean squared error                  0.2685
Relative absolute error                 56.5591 %
Root relative squared error             74.2526 %
Coverage of cases (0.95 level)          99.4485 %
Mean rel. region size (0.95 level)      72.2547 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.944    0.330    0.940      0.944    0.942      0.619    0.929     0.985     FALSE
                 0.670    0.056    0.686      0.670    0.678      0.619    0.929     0.737     TRUE
Weighted Avg.    0.901    0.288    0.900      0.901    0.901      0.619    0.929     0.947

=== Confusion Matrix ===

    a    b   <-- classified as
 9836  586 |    a = FALSE
  630 1278 |    b = TRUE
```

Figure.43. Results of the Random Forest penalizing false negative. Source: Weka.

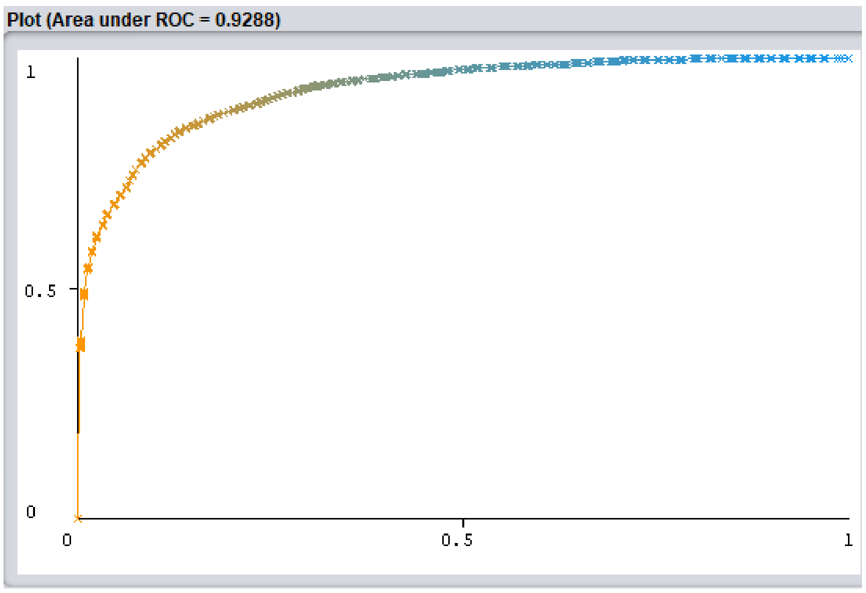


Figure.44. Roc plot of the Random Forest penalizing false negative. Source: Weka

Correctly Classified Instances (Accuracy) = 90.13 %

Incorrectly Classified Instances (Misclassification rate) = 9.86%

Roc = 0.9288

Total cost= 1216

False positive rate = 0.330

False negative rate = 0.056

False positive instances=630

False negative instances=586

We observe in the second model that correctly classified 90.13 % of the instances, whereas the misclassification was 9.86% and with a false positive rate of 0.330 and a false negative rate of 0.056. The total cost was 1216, which there were 630 False positive instances and 586 False negative instances, respectively. The Roc of the model was 0.9288.

8.2 INDIRECT METHOD - BOOSTING METHOD (ADDBOOSTM1)

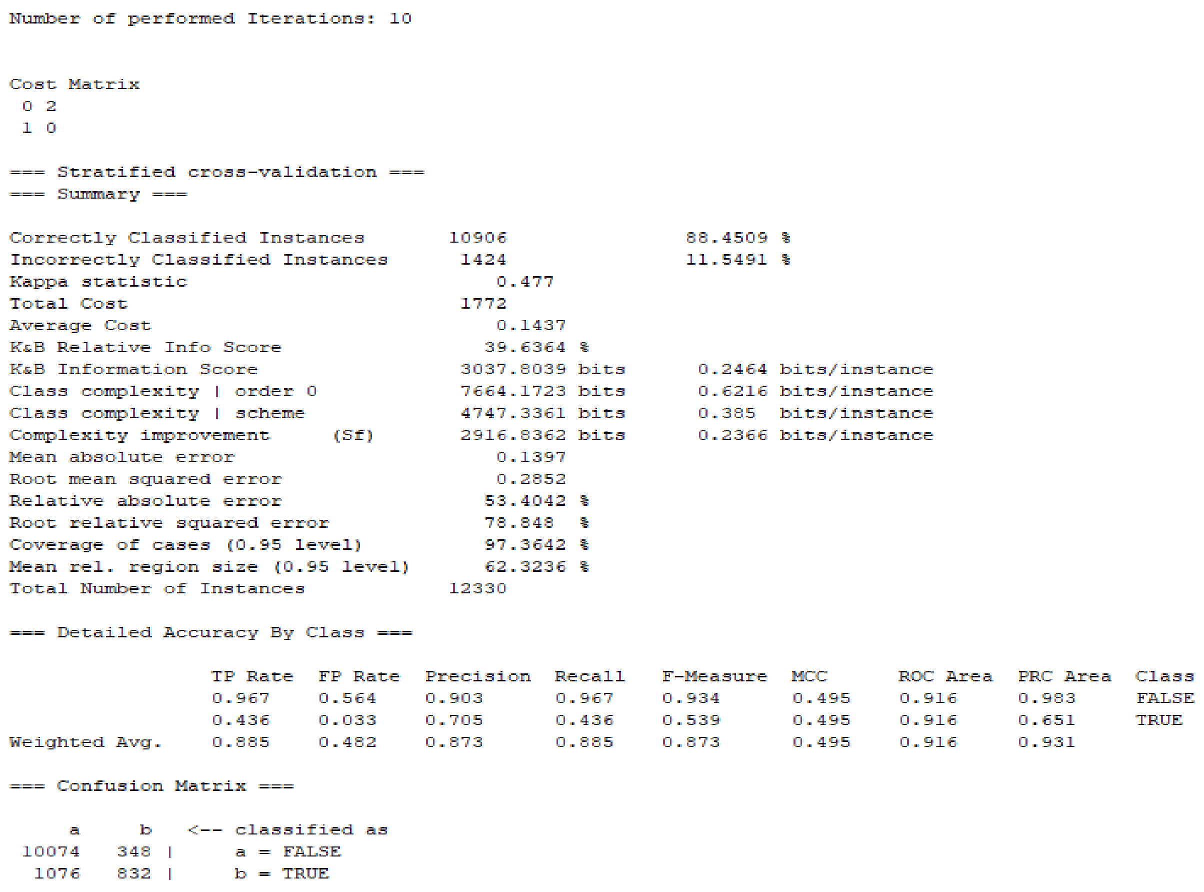
“Boosting is a Fmeta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones.”[25]

As we have seen in the background, we have three types of boosting algorithms AdaBoost, AdaCost and Weighting. In this section, we will analyse only the AdaBoost algorithm.

The AdaBoost algorithm [26] [27] aims to convert multiple weak classifiers into a single strong one. Essentially the AdaBoost are decision trees with a single split called decision stumps. The algorithm assigns more weight to the classifiers that perform poorly and less weight to the ones which are handled satisfactory well.

The AdaBoost models have been built using 66% training set and 33% test set, and a cross-validation 10-fold was used to train the model. The attribute used as a class label is revenue.

The first AdaBoost algorithm was build using a cost matrix, which penalized the false positive or customer that did not buy the product, but the model classed them as if they bought the product.



```
Number of performed Iterations: 10


Cost Matrix
 0 2
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10906               88.4509 %
Incorrectly Classified Instances      1424               11.5491 %
Kappa statistic                          0.477
Total Cost                            1772
Average Cost                             0.1437
K&B Relative Info Score                 39.6364 %
K&B Information Score                 3037.8039 bits      0.2464 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4747.3361 bits      0.385  bits/instance
Complexity improvement     (Sf)       2916.8362 bits      0.2366 bits/instance
Mean absolute error                      0.1397
Root mean squared error                  0.2852
Relative absolute error                 53.4042 %
Root relative squared error             78.848  %
Coverage of cases (0.95 level)          97.3642 %
Mean rel. region size (0.95 level)      62.3236 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.967    0.564    0.903      0.967    0.934      0.495    0.916     0.983     FALSE
                 0.436    0.033    0.705      0.436    0.539      0.495    0.916     0.651     TRUE
Weighted Avg.    0.885    0.482    0.873      0.885    0.873      0.495    0.916     0.931

=== Confusion Matrix ===

     a     b   <-- classified as
 10074   348 |     a = FALSE
  1076   832 |     b = TRUE
```

Figure.45. Results of the Boosting method (AddBoostm1) penalizing false positive. Source: Weka.

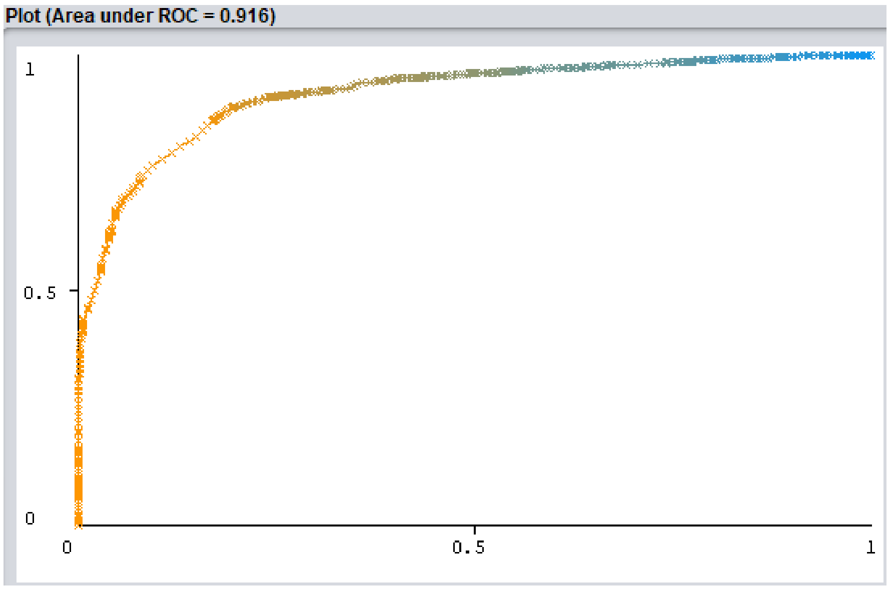


Figure.46. Roc plot of the Boosting method (AddBoostm1) penalizing false positive. Source: Weka

Correctly Classified Instances (Accuracy) = 88.45 %

Incorrectly Classified Instances (Misclassification rate) = 11.54%

Roc = 0.916

Total cost= 1772

False positive rate = 0.564

False negative rate = 0.033

False positive instances=1076

False negative instances=348

We see that the first AdaBoost algorithm correctly classified 88.45 % of the instances whereas the misclassification was 11.54% and with a false positive rate of 0.564 and a false negative rate of 0.033. The total cost was 1772, which there were 348 False positive instances and 1076 False negative instances, respectively.

The second AdaBoost algorithm was build using a cost matrix, which penalized false negative or the customer who bought the product, but the model classes them as if they did not buy.

```
Number of performed Iterations: 10


Cost Matrix
 0 1
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10848               87.9805 %
Incorrectly Classified Instances      1482               12.0195 %
Kappa statistic                          0.5889
Total Cost                            1948
Average Cost                             0.158
K&B Relative Info Score                 26.967  %
K&B Information Score                 2066.7998 bits      0.1676 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4880.0938 bits      0.3958 bits/instance
Complexity improvement     (Sf)       2784.0784 bits      0.2258 bits/instance
Mean absolute error                      0.1685
Root mean squared error                  0.289
Relative absolute error                 64.3857 %
Root relative squared error             79.9143 %
Coverage of cases (0.95 level)          99.262  %
Mean rel. region size (0.95 level)      73.6496 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.903    0.244    0.953      0.903    0.927      0.596    0.914     0.981     FALSE
                 0.756    0.097    0.587      0.756    0.661      0.596    0.914     0.660     TRUE
Weighted Avg.    0.880    0.222    0.896      0.880    0.886      0.596    0.914     0.931

=== Confusion Matrix ===

    a    b   <-- classified as
 9406 1016 |    a = FALSE
  466 1442 |    b = TRUE
```

Figure.47. Results of the Boosting method (AddBoostm1) penalizing false negative. Source: Weka.

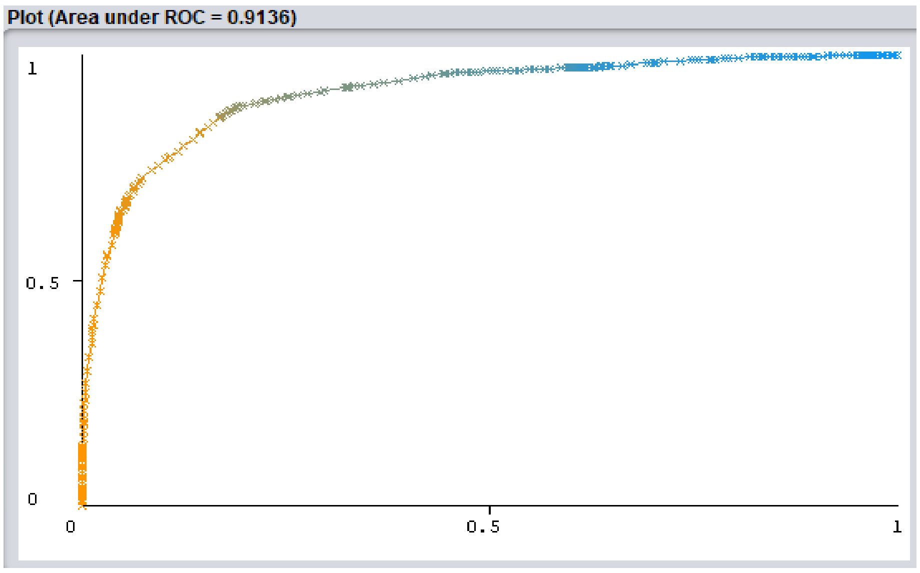


Figure.48. Roc plot of the Boosting method (AddBoostm1) penalizing false negative. Source: Weka

Correctly Classified Instances (Accuracy) = 87.98%

Incorrectly Classified Instances (Misclassification rate) = 12.01%

Roc= 0.9136

Total cost= 1948

False positive rate = 0.244

False negative rate = 0.097

False positive instances=466

False negative instances=1016

The second AdaBoost model classified 87.98% correctly of the instances whereas the misclassification was 12.01% and with a false positive rate of 0.244 and a false negative rate of 0.097. The total cost was 1948, which there were 466 False positive instances and 1016 False negative instances, respectively. The Roc of the model is 0.9136.

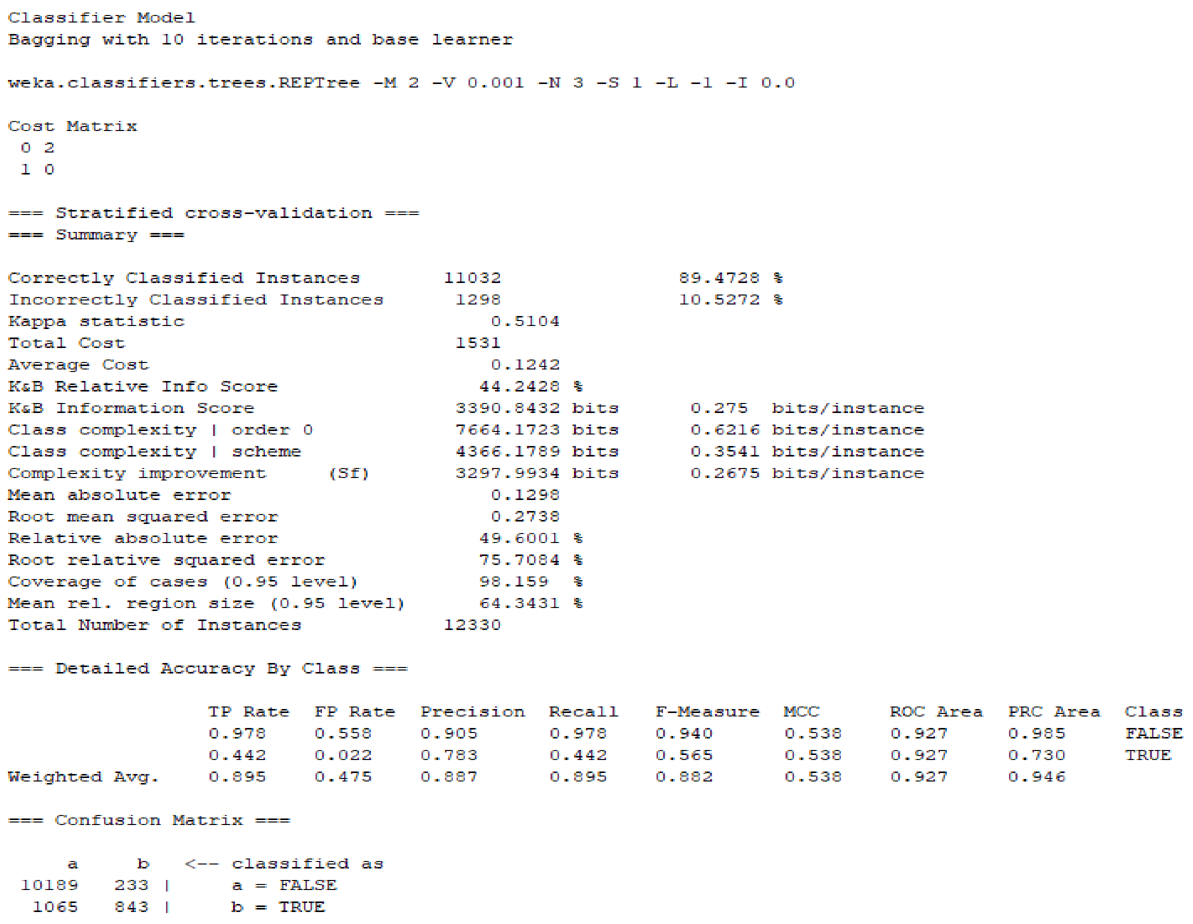
8.3 INDIRECT METHOD - BAGGING METHOD

"Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting."[28]

In order to avoid overfitting and reducing variance, the algorithm creates its variance by sampling and replacing data, while a model is tested. The models made up by the algorithm have the same weight, and voting is run to test what model is the is the most accurate.

Note that the bagging method was built using a Fast decision tree learner classifier and using 66% training set and 33% test set and a cross-validation 10-fold was used to train the model. The attribute used as a class label is revenue.

The first bagging model was build using a cost matrix, which penalized the false positive or customer that did not buy the product, but the model classed them as if they bought the product.



```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 2
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11032               89.4728 %
Incorrectly Classified Instances      1298               10.5272 %
Kappa statistic                          0.5104
Total Cost                            1531
Average Cost                             0.1242
K&B Relative Info Score                 44.2428 %
K&B Information Score                 3390.8432 bits      0.275  bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4366.1789 bits      0.3541 bits/instance
Complexity improvement     (Sf)       3297.9934 bits      0.2675 bits/instance
Mean absolute error                      0.1298
Root mean squared error                  0.2738
Relative absolute error                 49.6001 %
Root relative squared error             75.7084 %
Coverage of cases (0.95 level)          98.159  %
Mean rel. region size (0.95 level)      64.3431 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.978    0.558    0.905      0.978    0.940      0.538    0.927     0.985     FALSE
                 0.442    0.022    0.783      0.442    0.565      0.538    0.927     0.730     TRUE
Weighted Avg.    0.895    0.475    0.887      0.895    0.882      0.538    0.927     0.946

=== Confusion Matrix ===

     a     b   <-- classified as
 10189   233 |     a = FALSE
  1065   843 |     b = TRUE
```

Figure.49. Results of the Bagging method penalizing false positive. Source: Weka.

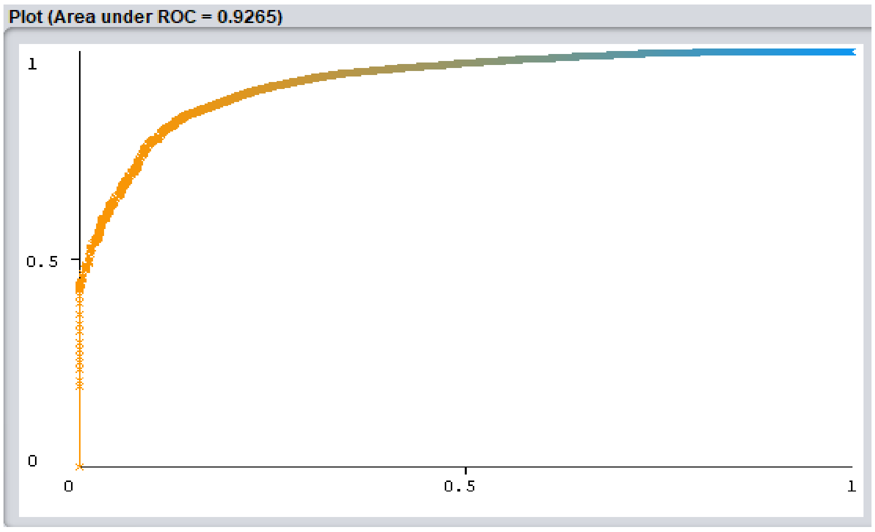


Figure.50. Roc plot of the Bagging method penalizing false positive. Source: Weka

Correctly Classified Instances (Accuracy) = 89.47 %

Incorrectly Classified Instances (Misclassification rate) = 10.52%

Total cost= 1531

Roc= 0.9265

False positive rate = 0.558

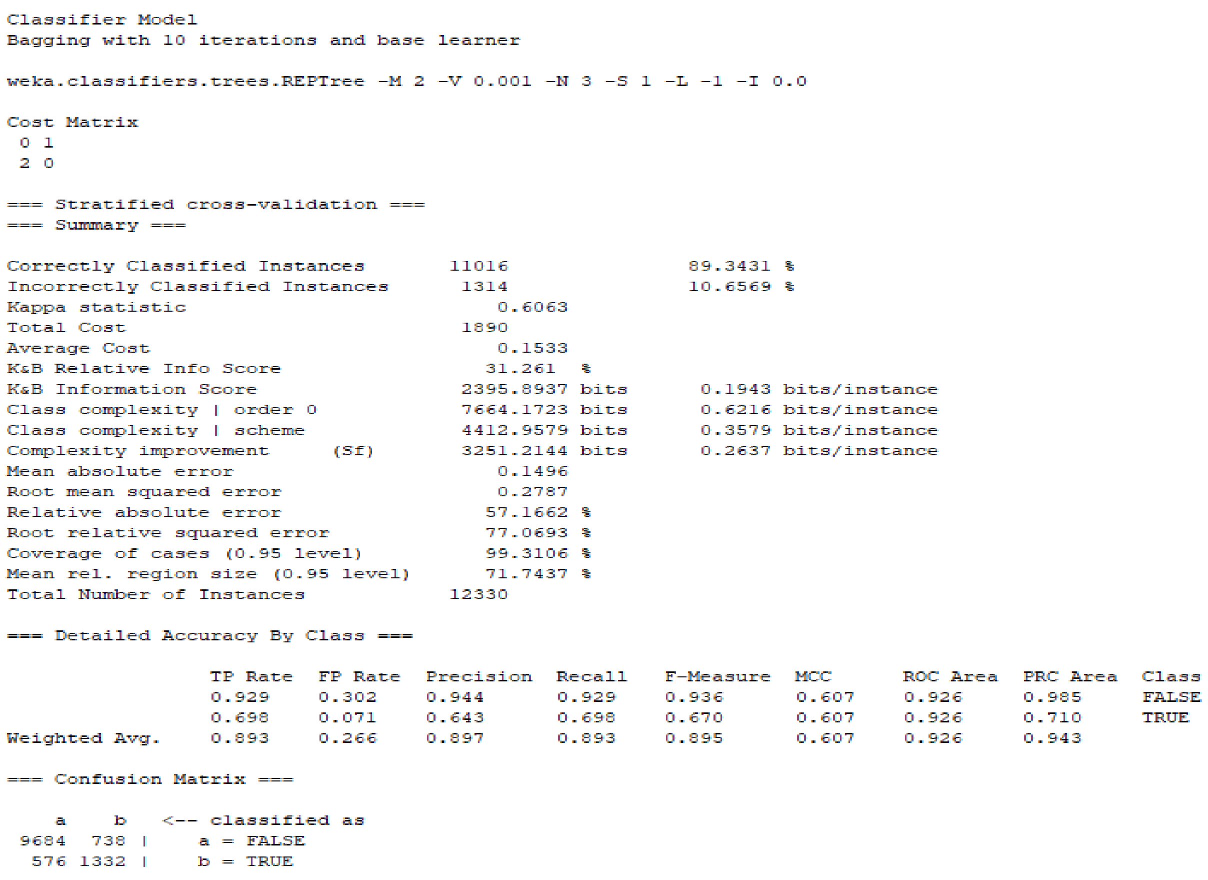
False negative rate = 0.022

False positive instances=1065

False negative instances=233

We see that the first bagging correctly classified 89.47 % of the instances whereas the misclassification was 10.52% and with a false positive rate of 0.558 and a false negative rate of 0.022. The total cost was 1531, which there were 1065 False positive instances and 233 False negative instances, respectively. The roc is 0.9265.

The second bagging model was build using penalized false negative or the customer who bought the product, but the model classes them as if they did not buy



```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 1
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11016               89.3431 %
Incorrectly Classified Instances      1314               10.6569 %
Kappa statistic                          0.6063
Total Cost                            1890
Average Cost                             0.1533
K&B Relative Info Score                 31.261  %
K&B Information Score                 2395.8937 bits      0.1943 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4412.9579 bits      0.3579 bits/instance
Complexity improvement      (Sf)      3251.2144 bits      0.2637 bits/instance
Mean absolute error                      0.1496
Root mean squared error                  0.2787
Relative absolute error                 57.1662 %
Root relative squared error             77.0693 %
Coverage of cases (0.95 level)          99.3106 %
Mean rel. region size (0.95 level)      71.7437 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.929    0.302    0.944      0.929    0.936      0.607    0.926     0.985     FALSE
                 0.698    0.071    0.643      0.698    0.670      0.607    0.926     0.710     TRUE
Weighted Avg.    0.893    0.266    0.897      0.893    0.895      0.607    0.926     0.943

=== Confusion Matrix ===

    a    b   <-- classified as
 9684  738 |    a = FALSE
  576 1332 |    b = TRUE
```

Figure.51. Results of the Bagging method penalizing false negative. Source: Weka.

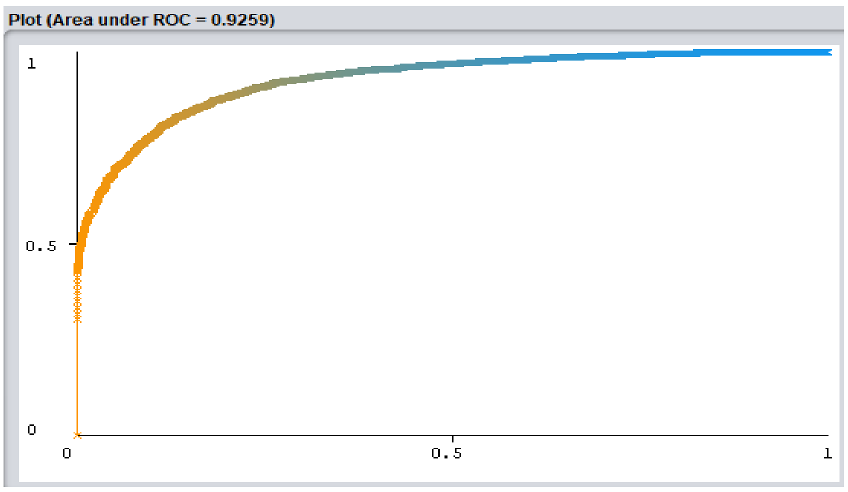


Figure.52. Roc plot of the Bagging method penalizing false negative. Source: Weka

Correctly Classified Instances (Accuracy) = 89.34 %

Incorrectly Classified Instances (Misclassification rate) = 10.65%

Roc= 0.9259

Total cost= 1890

False positive rate = 0.302

False positive rate = 0.071

False negative instances=576

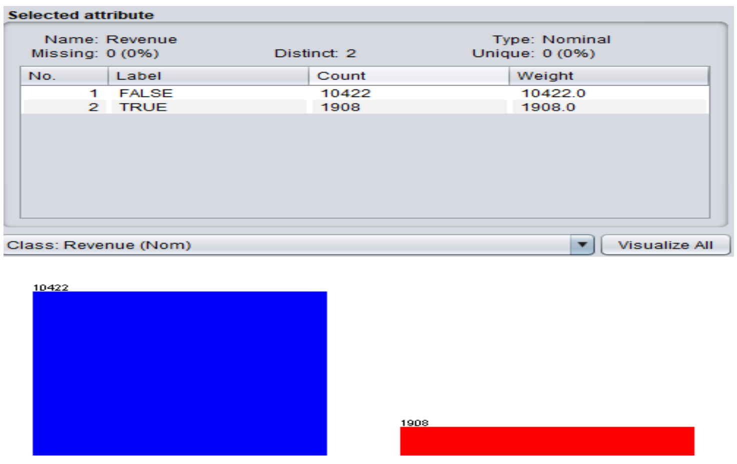
False negative instances=736

We see that the bagging model correctly classified 89.58 % of the instances whereas the misclassification was 10.41% and with a false positive rate of 0.278 and a false negative rate of 0.072. The total cost was 1284, which there were 754 False positive instances and 530 False negative instances, respectively. The Roc obtained is 0.9259.

8.4 INDIRECT METHOD - SAMPLING METHOD

The sampling method tackles the problem of imbalance data by modifying the number of rare cases in the training set regarding the cost [19].

This section we will build a model using the over-sampling method, which aims to increase the number of rare classes making the dataset more balanced and helping the algorithm to assign more weight to the rare instances. In order to illustrate how it works, we start recalling from the pre-processing section how unbalanced is the dataset, and in particular the revenue, which is considered the class label.


Selected attribute

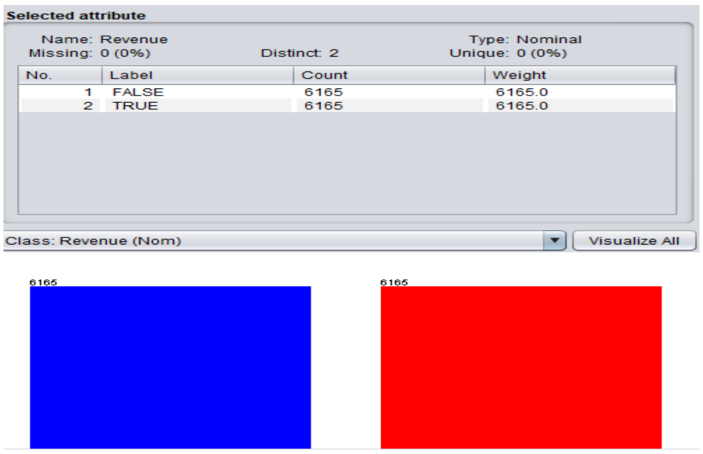
Name: Revenue
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	FALSE	10422	10422.0
2	TRUE	1908	1908.0

Class: Revenue (Nom)

Figure.53. Revenue attribute before performing over-sampling. Source: Weka.

After remembering the class label distribution, we can perform over-sampling on the dataset to be able to balance the data.


Selected attribute

Name: Revenue
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	FALSE	6165	6165.0
2	TRUE	6165	6165.0

Class: Revenue (Nom)

Figure.54. Revenue attribute after performing over-sampling. Source: Weka.

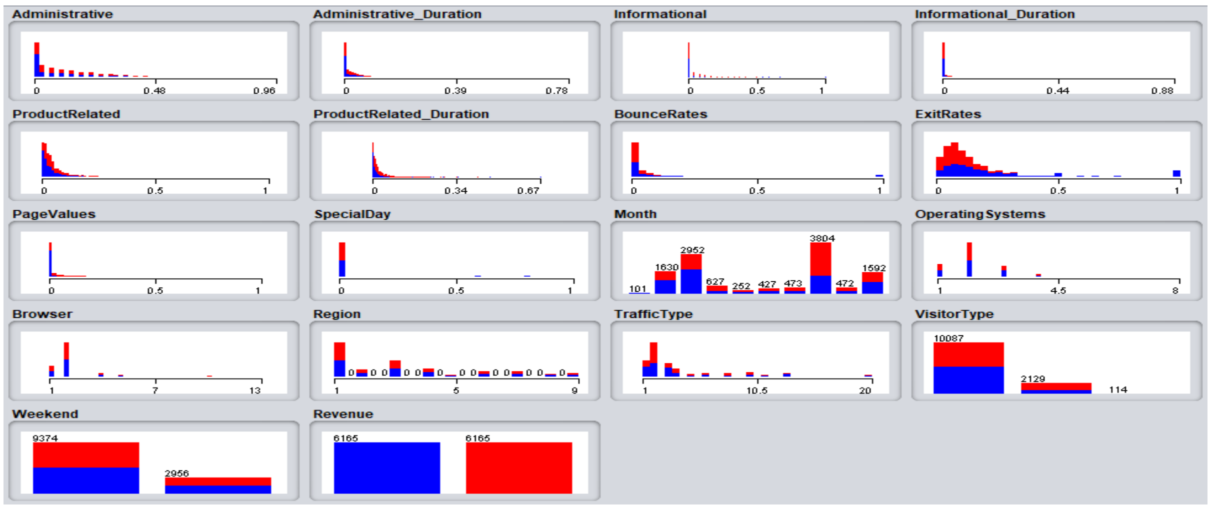
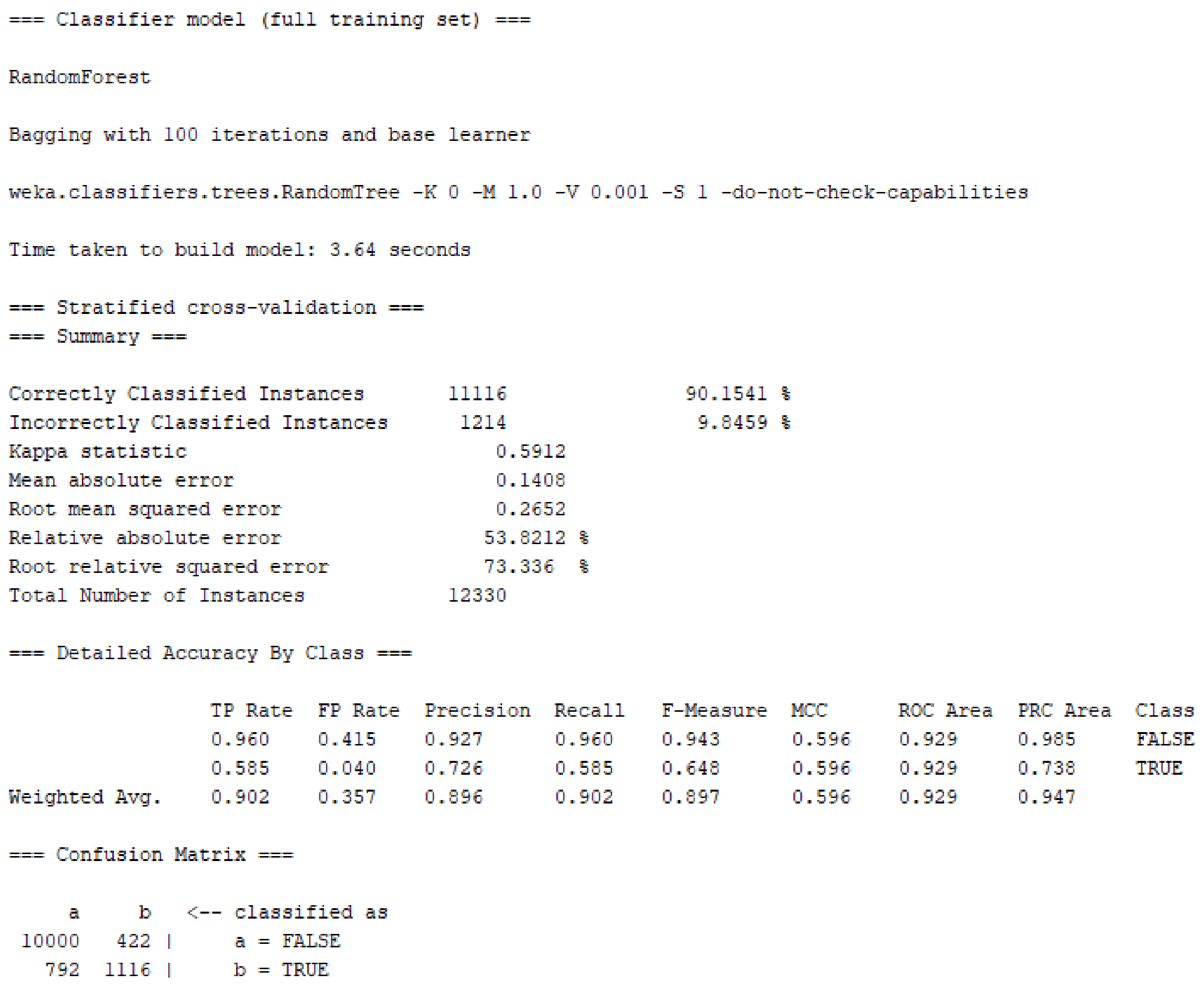


Figure.55. Overview of all the attributes after performing over-sampling. Source: Weka.

Now that the dataset has been over-sampled, we will build a Random Forest using the same parameters used at the beginning of this section when we built the direct method - Random Forest, and we will compare the model with a Random Forest with an original dataset with unbalance data.

The first Random Forest algorithm was built with the original data set with unbalanced data. Note that the Random Forest was built using 100 decision trees and using 66% training set and 33% test set and a cross validation 10-fold was used to train the model. The attribute used as a class label is revenue.



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

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: 3.64 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11116               90.1541 %
Incorrectly Classified Instances      1214                9.8459 %
Kappa statistic                          0.5912
Mean absolute error                      0.1408
Root mean squared error                  0.2652
Relative absolute error                 53.8212 %
Root relative squared error             73.336  %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.960    0.415    0.927      0.960    0.943      0.596    0.929     0.985     FALSE
                 0.585    0.040    0.726      0.585    0.648      0.596    0.929     0.738     TRUE
Weighted Avg.    0.902    0.357    0.896      0.902    0.897      0.596    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10000   422 |     a = FALSE
   792  1116 |     b = TRUE
```

Figure.56. Random Forest with Unbalanced data. Source: Weka.

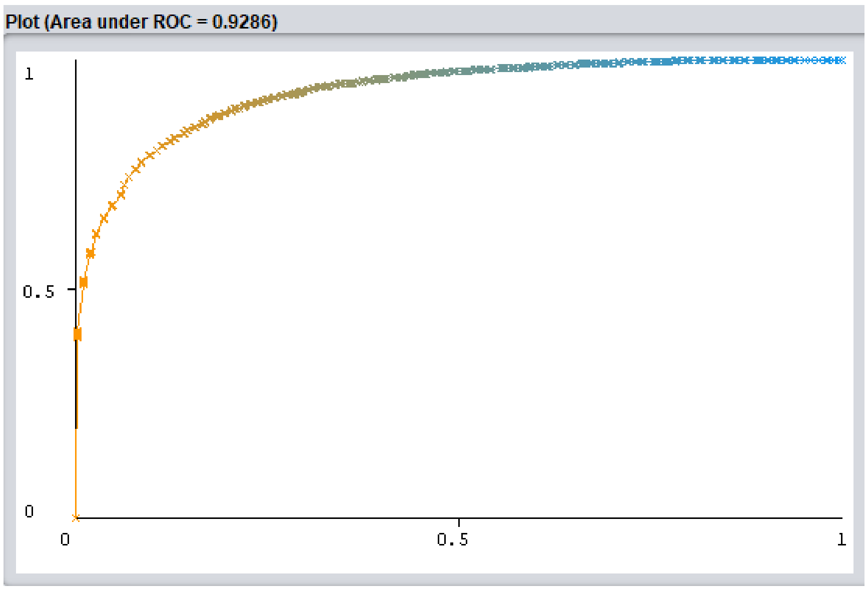


Figure.57. Roc plot of Random Forest with unbalanced data. Source: Weka

Correctly Classified Instances (Accuracy) = 90.15 %

Incorrectly Classified Instances (Misclassification rate) = 9.84%

Total cost= 1214

Roc= 0.9286

False positive rate = 0.415

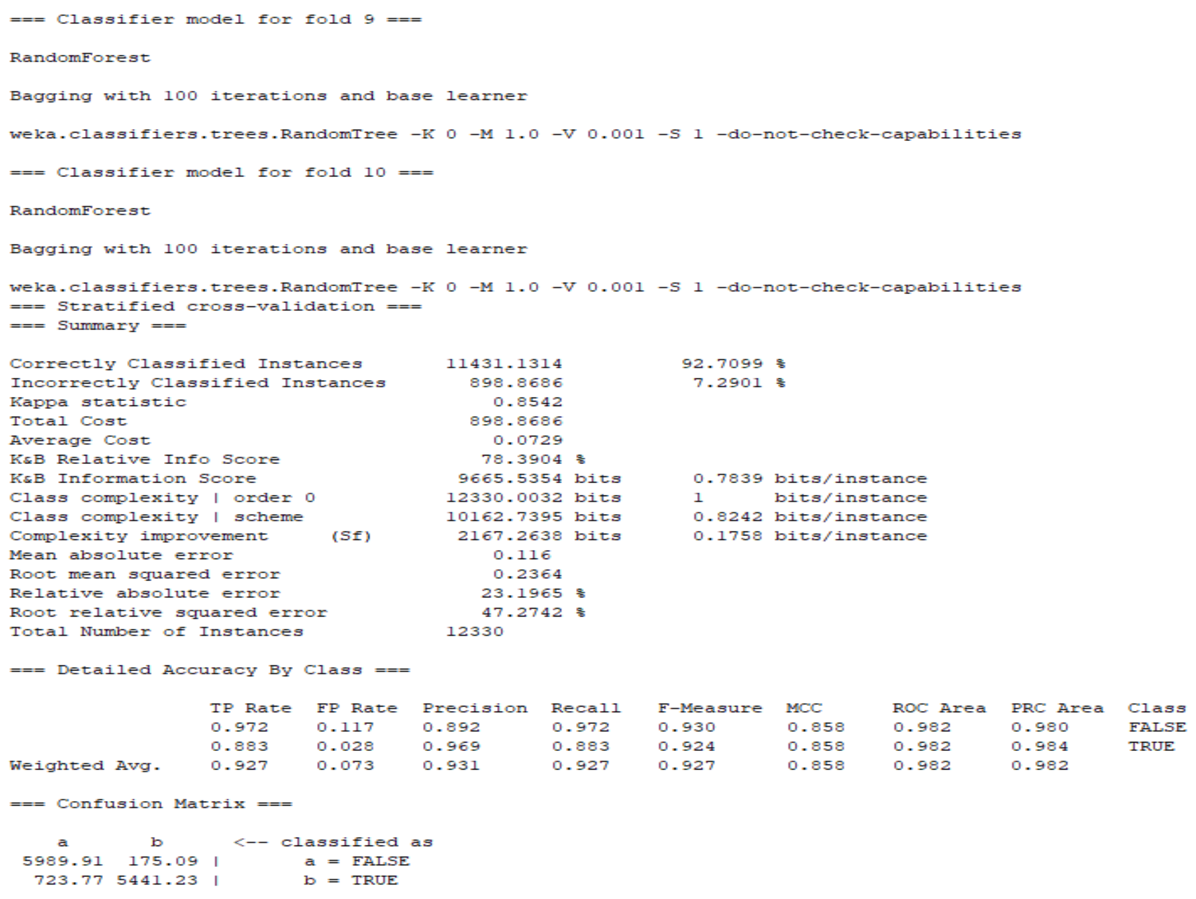
False negative rate = 0.040

False positive instances=792

False negative instances=442

We see that the first model with unbalanced dataset correctly classified 90.15 % of the instances whereas the misclassification was 9.84% and with a false positive rate of 0.415 and a false negative rate of 0.040. The total cost was 1214, which there were 792 False positive instances and 442 False negative instances, respectively. The Roc is 0.9286.

The second Random forest model was built using the over-sampling method to make the data to be balanced. Note that the Random Forest was built using 100 decision trees and using 66% training set and 33% test set and a cross-validation 10-fold was used to train the model. The attribute used as a class label is revenue.



```
=== Classifier model for fold 9 ===

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

=== Classifier model for fold 10 ===

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
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11431.1314          92.7099 %
Incorrectly Classified Instances       898.8686           7.2901 %
Kappa statistic                          0.8542
Total Cost                             898.8686
Average Cost                             0.0729
K&B Relative Info Score                 78.3904 %
K&B Information Score                 9665.5354 bits      0.7839 bits/instance
Class complexity | order 0           12330.0032 bits      1      bits/instance
Class complexity | scheme            10162.7395 bits      0.8242 bits/instance
Complexity improvement     (Sf)       2167.2638 bits      0.1758 bits/instance
Mean absolute error                      0.116
Root mean squared error                  0.2364
Relative absolute error                 23.1965 %
Root relative squared error             47.2742 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.972    0.117    0.892      0.972    0.930      0.858    0.982     0.980     FALSE
                 0.883    0.028    0.969      0.883    0.924      0.858    0.982     0.984     TRUE
Weighted Avg.    0.927    0.073    0.931      0.927    0.927      0.858    0.982     0.982

=== Confusion Matrix ===

    a       b      <-- classified as
 5989.91  175.09 |       a = FALSE
  723.77 5441.23 |       b = TRUE
```

Figure.58. Random Forest with balanced data. Source: Weka.

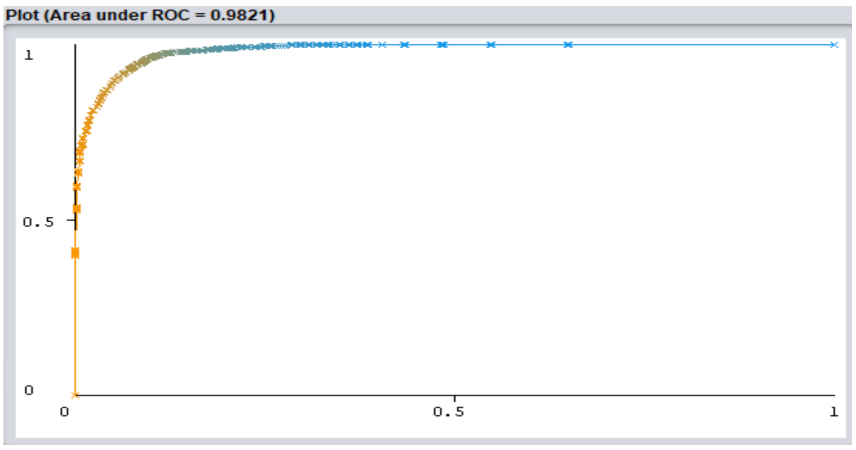


Figure.59. Roc plot of Random Forest with balanced data. Source: Weka

Correctly Classified Instances (Accuracy) = 92.70 %

Incorrectly Classified Instances (Misclassification rate) = 7.29%

Roc=0.9821

Total cost= 898

False positive rate = 0.117

False positive rate = 0.026

False negative instances=723

False negative instances=175

We see that the Random forest model built using the over-sampling method correctly classified 92.70 % of the instances whereas the misclassification was 7.59% and with a false positive rate of 0.117 and a false negative rate of 0.026 .The total cost was 898, which there were 723 False positive instances and 175 False negative instances, respectively. The Roc achieved is 0.9821.

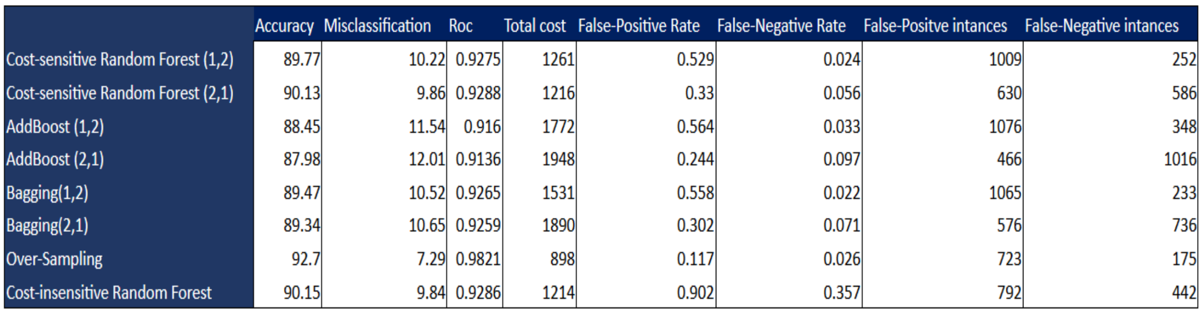
8.5 COST-SENSITIVE ALGORITHM EVALUATION

This section aims to compare the different models tested in regard to the online marketing domain, and therefore, to be able to use the most appropriate model according to the demands of the online marketing user.

First of all, to be able to see the entire picture and understanding why a model is more appropriate than other under certain circumstances we need to understand the impact of the different cost that our business suffers. In this particular case, we have two significant problems with the cost. The first cost is the one related to the false positive, which are the customers that did not buy the product, but the model classed them as if they bought the product. This cost makes the business to forecast more sales than it can sell and invest money in the customers who are not buying, this may lead to poor business decisions such as overstaffing, unwise investments, business profitability reduction and so on. The second cost is the one related false negative or the customer who bought the product, but the model classes them as if they did not buy. This cost also makes the business causes wrong forecasting estimating fewer sales than it can sell, it may lead in poor managerial decision and customers unsatisfied with the company due to understaffing and insufficient investment, loss of business reputation and so forth.

Both costs are important, and it is up to the user to decide which one is more important based on business needs and circumstances. In this section will explain what model suits the for every different sort of purpose, including the different types of costs.

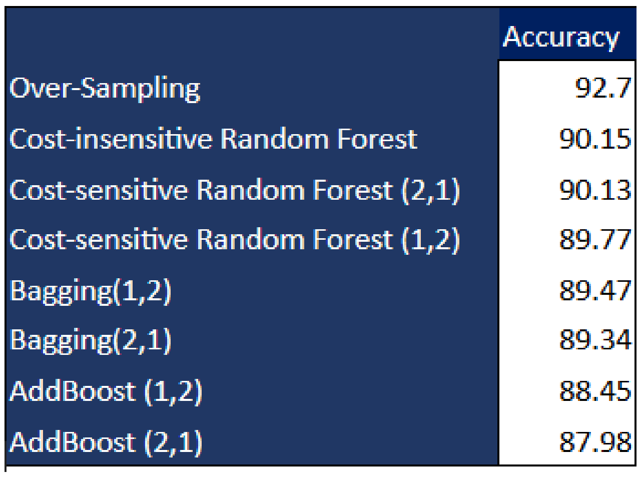
We will start gathering all the data extracted from the models and putting them all together to be able to compare and draw conclusions based on this information.



	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positve intances	False-Negative intances
Cost-sensitive Random Forest (1,2)	89.77	10.22	0.9275	1261	0.529	0.024	1009	252
Cost-sensitive Random Forest (2,1)	90.13	9.86	0.9288	1216	0.33	0.056	630	586
AddBoost (1,2)	88.45	11.54	0.916	1772	0.564	0.033	1076	348
AddBoost (2,1)	87.98	12.01	0.9136	1948	0.244	0.097	466	1016
Bagging(1,2)	89.47	10.52	0.9265	1531	0.558	0.022	1065	233
Bagging(2,1)	89.34	10.65	0.9259	1890	0.302	0.071	576	736
Over-Sampling	92.7	7.29	0.9821	898	0.117	0.026	723	175
Cost-insensitive Random Forest	90.15	9.84	0.9286	1214	0.902	0.357	792	442

Figure.60. Overview of the result of all the models tested. Source: Excel.

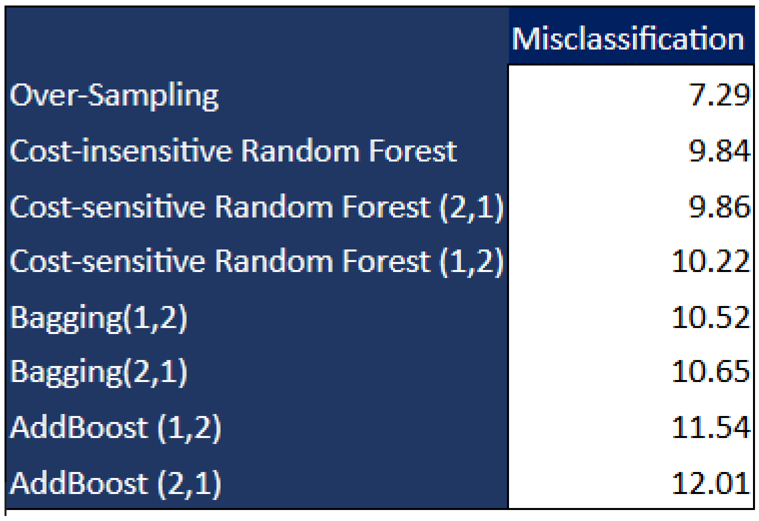
The data show that that best model in overall according to the accuracy, and not taking into consideration the costs, is the over-sampling method using Random Forest with an accuracy of 92.7%, followed by the insensitive Random Forest with a 90.15%. These results are not surprising since both algorithms are designed to maximize the accuracy in the models. Nevertheless, it is interesting to observe than the cost-sensitive algorithms score very well accuracy, and they are not distant from the cost-insensitive models.



	Accuracy
Over-Sampling	92.7
Cost-insensitive Random Forest	90.15
Cost-sensitive Random Forest (2,1)	90.13
Cost-sensitive Random Forest (1,2)	89.77
Bagging(1,2)	89.47
Bagging(2,1)	89.34
AddBoost (1,2)	88.45
AddBoost (2,1)	87.98

Figure.61. Accuracy results. Source: Excel.

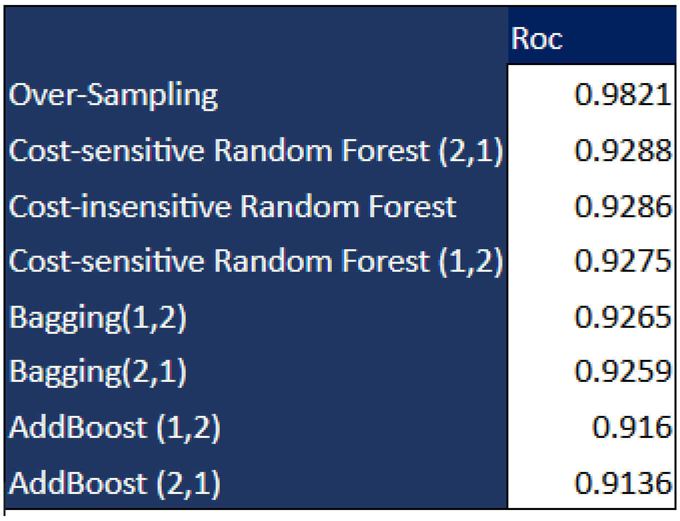
When It comes to the model misclassification, we expect the cost-sensitive algorithms to outperform the cost-insensitive algorithms. However, we find that the only cost-sensitive method that outperforms the cost-insensitive Random Forest is the Over-sampling method with 9.84% and 7.29%, respectively. The rest of the models have over 10% of misclassification, with the exception of the direct method 2 with 9.86%.



	Misclassification
Over-Sampling	7.29
Cost-insensitive Random Forest	9.84
Cost-sensitive Random Forest (2,1)	9.86
Cost-sensitive Random Forest (1,2)	10.22
Bagging(1,2)	10.52
Bagging(2,1)	10.65
AddBoost (1,2)	11.54
AddBoost (2,1)	12.01

Figure.62. Misclassification results. Source: Excel.

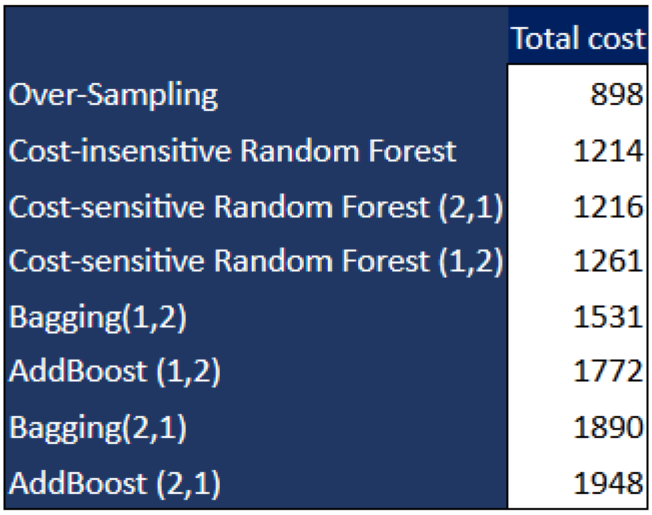
The Roc parameter shows that the best model is the Oversampling method with a 0.9821, whereas the worst performance is for the boosting method 2 with 0.9136.



	Roc
Over-Sampling	0.9821
Cost-sensitive Random Forest (2,1)	0.9288
Cost-insensitive Random Forest	0.9286
Cost-sensitive Random Forest (1,2)	0.9275
Bagging(1,2)	0.9265
Bagging(2,1)	0.9259
AddBoost (1,2)	0.916
AddBoost (2,1)	0.9136

Figure.63. Roc results. Source: Excel.

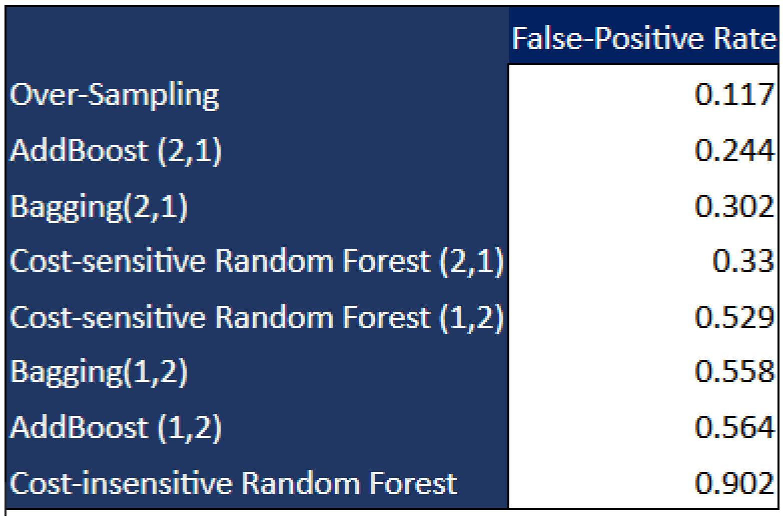
The total cost shows that the cost-sensitive oversampling method is the one with the least cost with 898 instances, followed by the insensitive Random Forest method with 1214 instances. On the contrary, the worst performance was the Boosting method 1 with 1948 instances.



	Total cost
Over-Sampling	898
Cost-insensitive Random Forest	1214
Cost-sensitive Random Forest (2,1)	1216
Cost-sensitive Random Forest (1,2)	1261
Bagging(1,2)	1531
AddBoost (1,2)	1772
Bagging(2,1)	1890
AddBoost (2,1)	1948

Figure.64. Total cost results. Source: Excel.

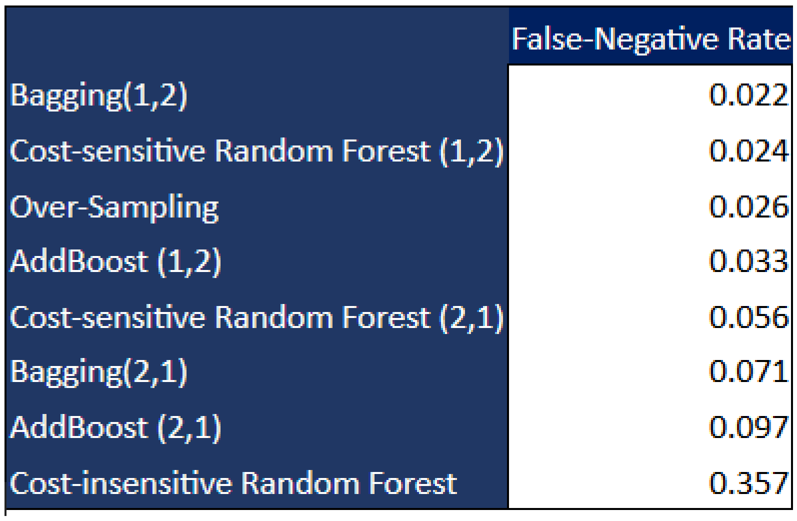
The False-positive rate is headed by the insensitive Oversampling with 0.117 and followed by the boosting and bagging method which penalized the false-negative with 0.244 and 0.302, respectively. On the other hand, the worst performances as expected are the algorithms that penalized the false-positive and the cost insensitive Random Forest.



	False-Positive Rate
Over-Sampling	0.117
AddBoost (2,1)	0.244
Bagging(2,1)	0.302
Cost-sensitive Random Forest (2,1)	0.33
Cost-sensitive Random Forest (1,2)	0.529
Bagging(1,2)	0.558
AddBoost (1,2)	0.564
Cost-insensitive Random Forest	0.902

Figure.65. False-positive results. Source: Excel.

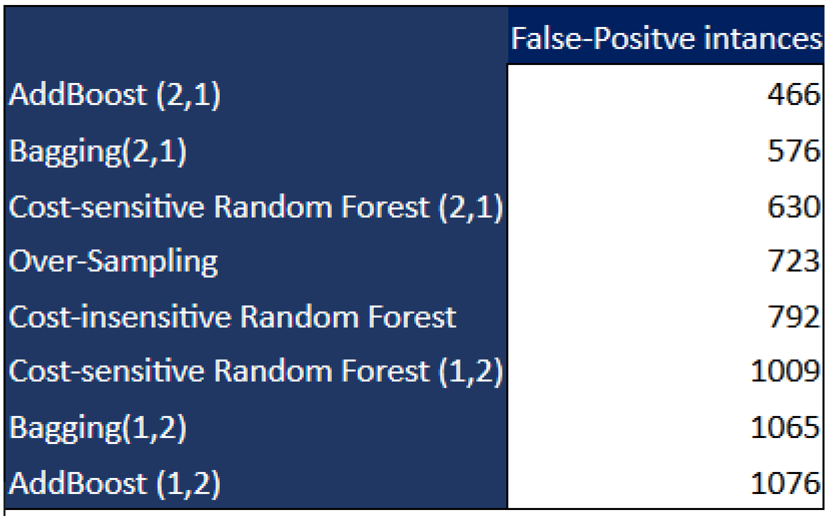
Regarding the false-negative rate, we find the opposite that we found on the false-positive rate. The models which penalized the false-positive outperformed all the models who penalized the false-negative. However, the best performer was the Oversampling method with 0.026, followed by Boosting method that penalized the false-positive with 0.097, whereas the worsts was the insensitive Random Forest and the methods that penalized the false-positive as expected.



	False-Negative Rate
Bagging(1,2)	0.022
Cost-sensitive Random Forest (1,2)	0.024
Over-Sampling	0.026
AddBoost (1,2)	0.033
Cost-sensitive Random Forest (2,1)	0.056
Bagging(2,1)	0.071
AddBoost (2,1)	0.097
Cost-insensitive Random Forest	0.357

Figure.66. False-negative results. Source: Excel.

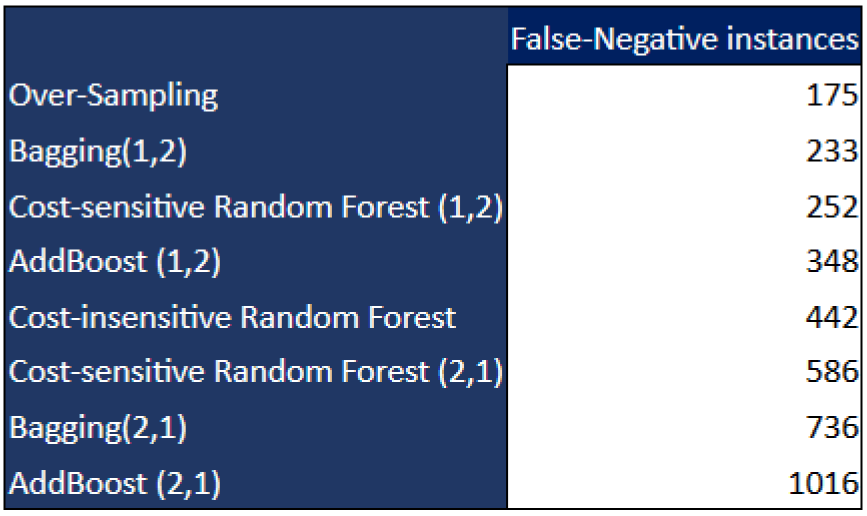
The models that had the least false-positive instances were the Boosting and Bagging methods that penalized false positive, whereas the one with most false-positive instances was the bagging method that penalized the false negative.



	False-Positve intances
AddBoost (2,1)	466
Bagging(2,1)	576
Cost-sensitive Random Forest (2,1)	630
Over-Sampling	723
Cost-insensitive Random Forest	792
Cost-sensitive Random Forest (1,2)	1009
Bagging(1,2)	1065
AddBoost (1,2)	1076

Figure.67. False-negative instances. Source: Excel.

The algorithms that had the least false-negative instances was the Over-Sampling method, whereas the one with most false-negative instances were methods that penalized the false-negative.



	False-Negative instances
Over-Sampling	175
Bagging(1,2)	233
Cost-sensitive Random Forest (1,2)	252
AddBoost (1,2)	348
Cost-insensitive Random Forest	442
Cost-sensitive Random Forest (2,1)	586
Bagging(2,1)	736
AddBoost (2,1)	1016

Figure.68. False-negative instances. Source: Excel.

We have seen how the different models tested during this project have performed under the different metrics and how all of them outstanding based on a particular characteristic.

The best model if we do not want to consider cost, and we focus on accuracy is the Oversampling methods. If we put the focus in the misclassification, the Oversampling method is the best model too. If we start considering the cost, but we do not make any distinction between cost we will look at the total cost, the best model is Oversampling method as well. If we consider that we need to minimize the false positive, we need to mix the information of false-positive instances and false positive rate and the best performer based on both metrics is the Boosting method that penalized the false-positive. Whereas if the goal is to reduce the false-negative, we need to mix information from false-negative instances and false-negative rate, and we conclude that the best model is the Oversampling method.

8.6- COST-SENSITIVE ALGORITHM - TESTING RESULTS

This section will test the results obtained in the previous sections to examine whether the cost caused the unsatisfactory performance of the bagging method, boosting method and the cost-sensitive random forest.

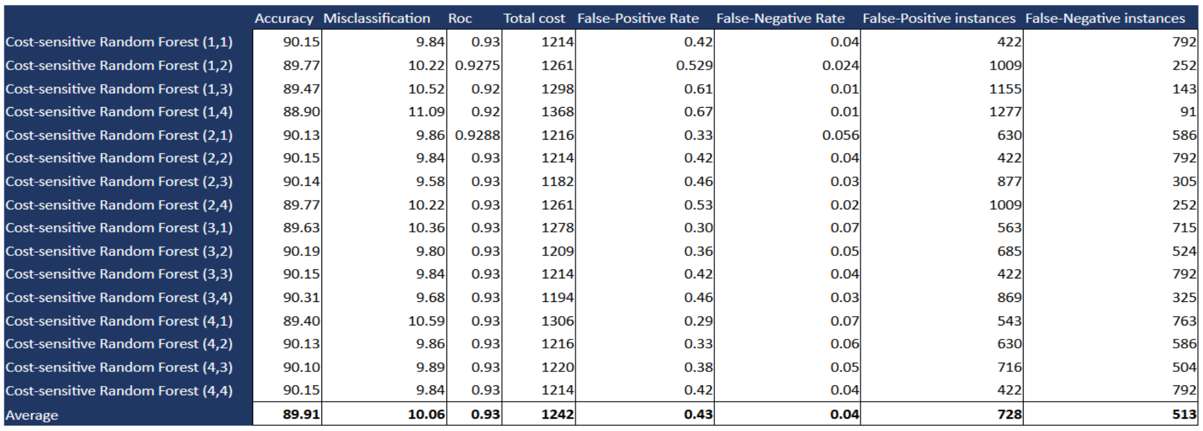
We will run different tests with different costs such as:

- Bagging method cost (FN, FP): (1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (2,3), (2,4), (3,1),(3,2), (3,3), (3,4), (4,1), (4,2), (4,3), (4,4)
- Boosting method (FN, FP):): (1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (2,3), (2,4), (3,1),(3,2), (3,3), (3,4), (4,1), (4,2), (4,3), (4,4)
- Cost-sensitive random forest (FN, FP): (): (1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (2,3), (2,4), (3,1),(3,2), (3,3), (3,4), (4,1), (4,2), (4,3), (4,4)

The result will be averaged and compared against the oversampling method.

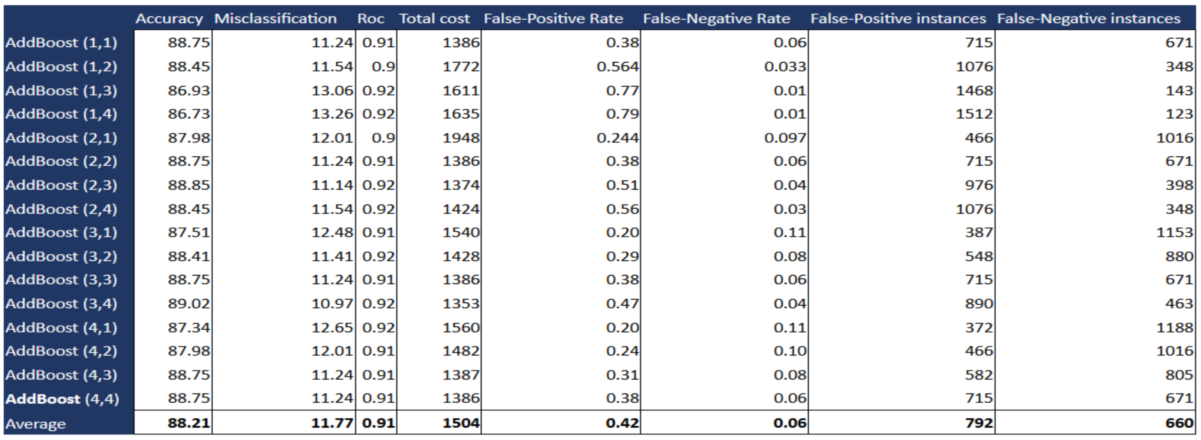
Note that the different model will be available in the appendix. This section will contain tables with the information of the models.

The following tables are the cost-sensitive Random forest, AddBoost and Bagging models and its different metrics to be able to assess them. The cost changes from 1 to 4 for the false negative and false positive.



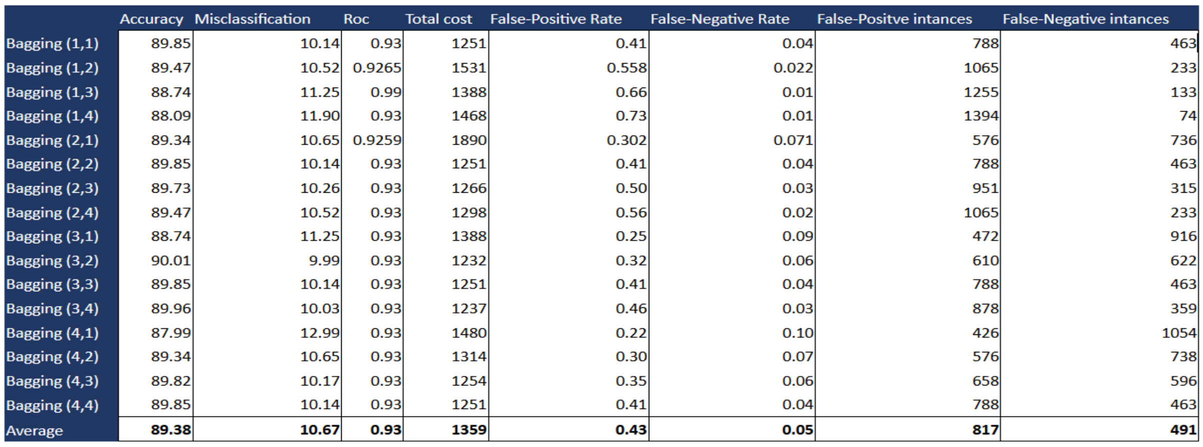
	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positive instances	False-Negative instances
Cost-sensitive Random Forest (1,1)	90.15	9.84	0.93	1214	0.42	0.04	422	792
Cost-sensitive Random Forest (1,2)	89.77	10.22	0.9275	1261	0.529	0.024	1009	252
Cost-sensitive Random Forest (1,3)	89.47	10.52	0.92	1298	0.61	0.01	1155	143
Cost-sensitive Random Forest (1,4)	88.90	11.09	0.92	1368	0.67	0.01	1277	91
Cost-sensitive Random Forest (2,1)	90.13	9.86	0.9288	1216	0.33	0.056	630	586
Cost-sensitive Random Forest (2,2)	90.15	9.84	0.93	1214	0.42	0.04	422	792
Cost-sensitive Random Forest (2,3)	90.14	9.58	0.93	1182	0.46	0.03	877	305
Cost-sensitive Random Forest (2,4)	89.77	10.22	0.93	1261	0.53	0.02	1009	252
Cost-sensitive Random Forest (3,1)	89.63	10.36	0.93	1278	0.30	0.07	563	715
Cost-sensitive Random Forest (3,2)	90.19	9.80	0.93	1209	0.36	0.05	685	524
Cost-sensitive Random Forest (3,3)	90.15	9.84	0.93	1214	0.42	0.04	422	792
Cost-sensitive Random Forest (3,4)	90.31	9.68	0.93	1194	0.46	0.03	869	325
Cost-sensitive Random Forest (4,1)	89.40	10.59	0.93	1306	0.29	0.07	543	763
Cost-sensitive Random Forest (4,2)	90.13	9.86	0.93	1216	0.33	0.06	630	586
Cost-sensitive Random Forest (4,3)	90.10	9.89	0.93	1220	0.38	0.05	716	504
Cost-sensitive Random Forest (4,4)	90.15	9.84	0.93	1214	0.42	0.04	422	792
Average	**89.91**	**10.06**	**0.93**	**1242**	**0.43**	**0.04**	**728**	**513**

Figure.69. Cost-sensitive Random Forest models and its average. Source: Excel.



	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positive instances	False-Negative instances
AddBoost (1,1)	88.75	11.24	0.91	1386	0.38	0.06	715	671
AddBoost (1,2)	88.45	11.54	0.9	1772	0.564	0.033	1076	348
AddBoost (1,3)	86.93	13.06	0.92	1611	0.77	0.01	1468	143
AddBoost (1,4)	86.73	13.26	0.92	1635	0.79	0.01	1512	123
AddBoost (2,1)	87.98	12.01	0.9	1948	0.244	0.097	466	1016
AddBoost (2,2)	88.75	11.24	0.91	1386	0.38	0.06	715	671
AddBoost (2,3)	88.85	11.14	0.92	1374	0.51	0.04	976	398
AddBoost (2,4)	88.45	11.54	0.92	1424	0.56	0.03	1076	348
AddBoost (3,1)	87.51	12.48	0.91	1540	0.20	0.11	387	1153
AddBoost (3,2)	88.41	11.41	0.92	1428	0.29	0.08	548	880
AddBoost (3,3)	88.75	11.24	0.91	1386	0.38	0.06	715	671
AddBoost (3,4)	89.02	10.97	0.92	1353	0.47	0.04	890	463
AddBoost (4,1)	87.34	12.65	0.92	1560	0.20	0.11	372	1188
AddBoost (4,2)	87.98	12.01	0.91	1482	0.24	0.10	466	1016
AddBoost (4,3)	88.75	11.24	0.91	1387	0.31	0.08	582	805
AddBoost (4,4)	88.75	11.24	0.91	1386	0.38	0.06	715	671
Average	**88.21**	**11.77**	**0.91**	**1504**	**0.42**	**0.06**	**792**	**660**

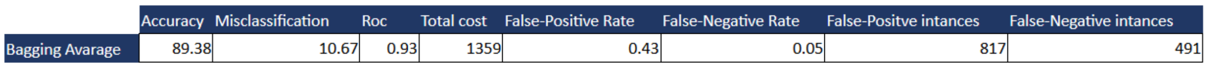
Figure.70. AddBoost models and its average. Source: Excel.



	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positve intances	False-Negative intances
Bagging (1,1)	89.85	10.14	0.93	1251	0.41	0.04	788	463
Bagging (1,2)	89.47	10.52	0.9265	1531	0.558	0.022	1065	233
Bagging (1,3)	88.74	11.25	0.99	1388	0.66	0.01	1255	133
Bagging (1,4)	88.09	11.90	0.93	1468	0.73	0.01	1394	74
Bagging (2,1)	89.34	10.65	0.9259	1890	0.302	0.071	576	736
Bagging (2,2)	89.85	10.14	0.93	1251	0.41	0.04	788	463
Bagging (2,3)	89.73	10.26	0.93	1266	0.50	0.03	951	315
Bagging (2,4)	89.47	10.52	0.93	1298	0.56	0.02	1065	233
Bagging (3,1)	88.74	11.25	0.93	1388	0.25	0.09	472	916
Bagging (3,2)	90.01	9.99	0.93	1232	0.32	0.06	610	622
Bagging (3,3)	89.85	10.14	0.93	1251	0.41	0.04	788	463
Bagging (3,4)	89.96	10.03	0.93	1237	0.46	0.03	878	359
Bagging (4,1)	87.99	12.99	0.93	1480	0.22	0.10	426	1054
Bagging (4,2)	89.34	10.65	0.93	1314	0.30	0.07	576	738
Bagging (4,3)	89.82	10.17	0.93	1254	0.35	0.06	658	596
Bagging (4,4)	89.85	10.14	0.93	1251	0.41	0.04	788	463
Average	**89.38**	**10.67**	**0.93**	**1359**	**0.43**	**0.05**	**817**	**491**

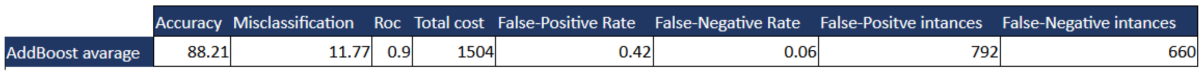
Figure.71. Bagging models and its average. Source: Excel.

Now that we have all the information from the model, we will put the results together, and we compare it to conclude what is the best model for working with the Online Marketing Dataset.



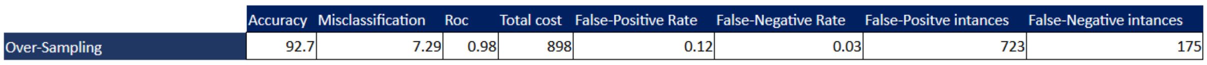
	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positve intances	False-Negative intances
Bagging Avarage	89.38	10.67	0.93	1359	0.43	0.05	817	491

Figure.72. Average of the Bagging models. Source: Excel.



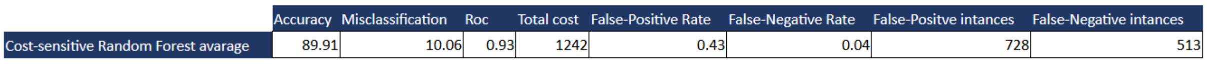
	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positve intances	False-Negative intances
AddBoost avarage	88.21	11.77	0.9	1504	0.42	0.06	792	660

Figure.73. Average of the AddBoost models. Source: Excel.



	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positve intances	False-Negative intances
Over-Sampling	92.7	7.29	0.98	898	0.12	0.03	723	175

Figure.74. Average of the Over-sampling model. Source: Excel.



	Accuracy	Misclassification	Roc	Total cost	False-Positive Rate	False-Negative Rate	False-Positve intances	False-Negative intances
Cost-sensitive Random Forest avarage	89.91	10.06	0.93	1242	0.43	0.04	728	513

Figure.75. Average of the Cost-Sensitive Random Forest model. Source: Excel.

As we can observe from the results of the models obtained after averaging the models with the different costs, we can see that the Over-sampling still being the technique that performs better on the Online Marketing Dataset, since it gives the best accuracy and the minimum cost. It is also important to mention that in any model tested the results were better.

9 CRITICAL EVALUATION

The use of cost-sensitive algorithms in Online Marketing project has covered the most relevant literature extensively on cost-sensitive algorithms, its application in marketing and its theory. Besides, the project has used an Online Marketing dataset to deploy the four main categories of cost-sensitive algorithms such as the direct method, boosting, bagging and sampling method and the results have been analysed and compared under the cost viewpoint.

In the introduction of cost-sensitive algorithms was introduced the most relevant literature on cost-sensitive, the types of cost-sensitive algorithms and the most relevant research done in the field of Online Marketing deploying cost-sensitive algorithms. Besides, the project introduces the types of cost and overview of the theory of the cost-sensitive algorithm.

The project showed how the Online Marketing dataset was pre-processed in order to deploy the cost-sensitive algorithms. The project tested five different cost-sensitive approaches and we run over 50 models.

Finally, the project compared the results of the algorithms in the Online Marketing dataset and made sense of them under the cost viewpoint.

The results have shown that there is not a perfect algorithm that works for all purposes. However, we found that some methods such as over-sampling work better than others in most of the cases for the Online Marketing Dataset. Therefore, it is up to the user to assess the type of cost to be able to identify what model fits better regarding business needs.

In this regard, one of the lessons learned from this project is that sometimes simple methods such as the over-sampling methods work better than other more complex ones. Moreover, certain imbalanced data sets, such as the one used in the project, are not appropriately handled by the cost-sensitive algorithms because they are not made to handle imbalanced class distributions.

10 CONCLUSION

The use of cost-sensitive algorithms in Online Marketing covered the most relevant literature and theory on cost-sensitive algorithms. The project performed tests in the main methods within the cost-sensitive field and compared the results under the cost viewpoint.

The findings in the project are very similar to the one published in the paper Evaluation of Cost-Sensitive Learning for Imbalanced Bank Direct Marketing Data [7], even though the dataset was different and instead of online marketing the research was performed in direct marketing of a bank. In general, the best method to tackle imbalanced data is over-sampling as the results have demonstrated. However, in this project, as opposed to Evaluation of Cost-Sensitive Learning for Imbalanced Bank Direct Marketing Data paper, I consider that every business needs to assess the type of cost and how it is affecting and then select the best model that in many cases will be over-sampling but not in all cases.

On the other hand, a possible improvement to the project and further work could include and optimisation of the parameters on the algorithms and perform more test in more datasets to be able to compare results and see if the over-sampling method is consistently the best model for marketing datasets.

REFERENCES

1- Turney.P. Types of cost in inductive concept learning. Institute for information technology of the national research council of Canada Ottawa, Canada, 2000.

2- Turney.P. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision. Journal of artificial intelligence research, 1995

3- Qin, Z., Zhang, C., Wang, T., & Zhang, S. Cost sensitive classification in data mining. In Advanced Data Mining and Applications, 2011.

4- Zadrozny, B., Langford, J., & Abe, N. Cost-sensitive learning by cost-proportionate example weighting. Data Mining. Third IEEE International Conference, 2003.

5- Elkan, C. The foundations of cost-sensitive learning. International joint conference on artificial intelligence.2005

6- Shilbayeh.S. Cost Sensitive meta learning. University of Salford, 2011.

7- Khor Kok-Chin and Ng Keng-Hoong. Evaluation of Cost Sensitive Learning for Imbalanced Bank Direct Marketing Data. Indian Journal of Science and Technology.2016

8- Wikipedia Contributors (2019). Data Mining.[Online]. Wikipedia Available at: https://en.wikipedia.org/wiki/Thanos [Accessed 05 Jul. 2019].

9- M Saraee. Data Mining notes. University of Salford, 2018.

10- Zadrozny, B. and Elkan, C. Learning and making decisions when costs and probabilities are both Unknown. Seventh International Conference on Knowledge Discovery and Data Mining. 2001.

11- Elkan, C. The Foundations of Cost-Sensitive Learning. the Seventeenth International Joint Conference of Artificial Intelligence. 2001.

12- Ling, C.X., Yang, Q., Wang, J., and Zhang, S. Decision Trees with Minimal Costs. Proceedings of 2004 International Conference on Machine Learning.2011

13- Schapire & Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. AT&T Labs, Shannon Laboratory,1999.

14- Freund, Seung, Shamir, & Tishby. Experiments with a New Boosting Algorithm.Machine Learning: Proceedings of the Thirteenth International Conference.1993.

15- Fan, Stolfo, Zhang, & Chan. AdaCost : Misclassification Cost-sensitive Boosting. 1999

16- Ting, K.M. Inducing Cost-Sensitive Trees via Instance Weighting. In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery.1998.

17- Domingos, P. MetaCost: A general method for making classifiers cost sensitive. Proceedings of the Fifth International Conference on Knowledge Discovery and Data.1999

18- Zadrozny, B., Langford, J., and Abe, N. Cost-sensitive learning by Cost-Proportionate instance Weighting. In Proceedings of the 3th International Conference on Data Mining.2003

19- Elkan. The Foundations of Cost-Sensitive Learning. Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence.2001.

20- Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research.2002.

21- Sakar, C.O., Polat, S.O., Katircioglu, M. et al. Neural Comput & Applic .2018

22- Uci (Machine learning repository),2019. Uci Available at https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset.

23- Dr. Judita Preiss, Charith Silva. ASDM Workshop: Normalization. Salford University.2019.

24- Wikipedia Contributors (2019). Radom Forest. [Online]. https://en.wikipedia.org/wiki/Random_forest [Accessed 06 Aug. 2019].

25- Wikipedia Contributors (2019). Boosting (Machine Learning). [Online]. https://en.wikipedia.org/wiki/Boosting_(machine_learning) [Accessed 06 Aug. 2019].

26- Akash Desarda .(Jul 17, 2019).Understanding AdaBoost. https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe.

27- SauceCat. (Apr 29, 2017). Boosting algorithm: AdaBoost. https://towardsdatascience.com/boosting-algorithm-adaboost-b6737a9ee60c.

28- Wikipedia Contributors (2019). Bootstrap aggregating. [Online]. https://en.wikipedia.org/wiki/Bootstrap_aggregating [Accessed 06 Aug. 2019].

29- E Nashnush, S Vadera. Learning cost-sensitive Bayesian networks via direct and indirect methods. Integrated Computer-Aided Engineering 24 (1), 17-26. Salford University.

30- E Nashnush, S Vadera. Cost-sensitive Bayesian network learning using sampling. Recent Advances on Soft Computing and Data Mining, 467-476. Salford University.

31- E Nashnush. Development of new cost-sensitive Bayesian network learning algorithms. Salford university.

APPENDIX

- **R code to clean data in section 7 (Data processing)**

```
setwd("c:/Users/carlo/Desktop/Project Cost-sensitive")

mrktdata<-read.csv('Marketing Dataset.csv',header = T)

##INITIAL EXPLORATION OF DATA ##

str(mrktdata)

summary(mrktdata$Region)

##NORMALIZING ATTRIBUTES##

mrktdata$Administrative<- (mrktdata$Administrative-
min(mrktdata$Administrative))/(max(mrktdata$Administrative)-min(mrktdata$Administrative))

mrktdata$Administrative_Duration<- (mrktdata$Administrative_Duration-
min(mrktdata$Administrative_Duration))/(max(mrktdata$Administrative_Duration)-
min(mrktdata$Administrative_Duration))

mrktdata$Informational<- (mrktdata$Informational-
min(mrktdata$Informational))/(max(mrktdata$Informational)-min(mrktdata$Informational))

mrktdata$Informational_Duration<- (mrktdata$Informational_Duration-
min(mrktdata$Informational_Duration))/(max(mrktdata$Informational_Duration)-
min(mrktdata$Informational_Duration))

mrktdata$ProductRelated<- (mrktdata$ProductRelated-
min(mrktdata$ProductRelated))/(max(mrktdata$ProductRelated)-min(mrktdata$ProductRelated))

mrktdata$ProductRelated_Duration<- (mrktdata$ProductRelated_Duration-
min(mrktdata$ProductRelated_Duration))/(max(mrktdata$ProductRelated_Duration)-
min(mrktdata$ProductRelated_Duration))

mrktdata$BounceRates<- (mrktdata$BounceRates-min(mrktdata$BounceRates))/(max(mrktdata$BounceRates)-
min(mrktdata$BounceRates))

mrktdata$ExitRates<- (mrktdata$ExitRates-min(mrktdata$ExitRates))/(max(mrktdata$ExitRates)-
min(mrktdata$ExitRates))

mrktdata$PageValues<- (mrktdata$PageValues-min(mrktdata$PageValues))/(max(mrktdata$PageValues)-
min(mrktdata$PageValues))

##SWAPING ATTRIBUTES FROM NUMERICAL TO FACTORIAL ##

mrktdata$SpecialDay<-as.factor(mrktdata$SpecialDay)

mrktdata$OperatingSystems<-as.factor(mrktdata$OperatingSystems)

mrktdata$Browser<-as.factor(mrktdata$Browser)

mrktdata$Region<-as.factor(mrktdata$Region)
```

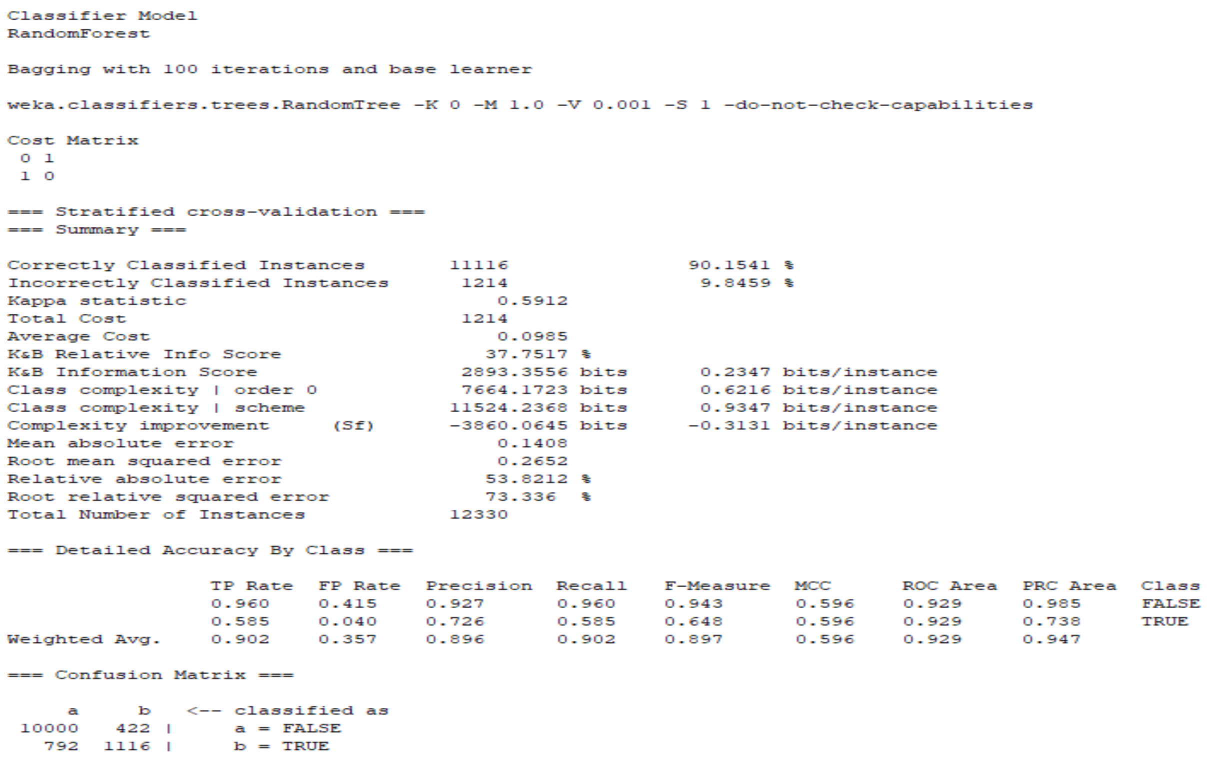
```
mrktdata$TrafficType<-as.factor(mrktdata$TrafficType)

mrktdata$Weekend<-as.factor(mrktdata$Weekend)

mrktdata$Revenue<-as.factor(mrktdata$Revenue)
```

##RESULTS OF DATA CLEANING ##

```
str(mrktdata)

summary(mrktdata)
```

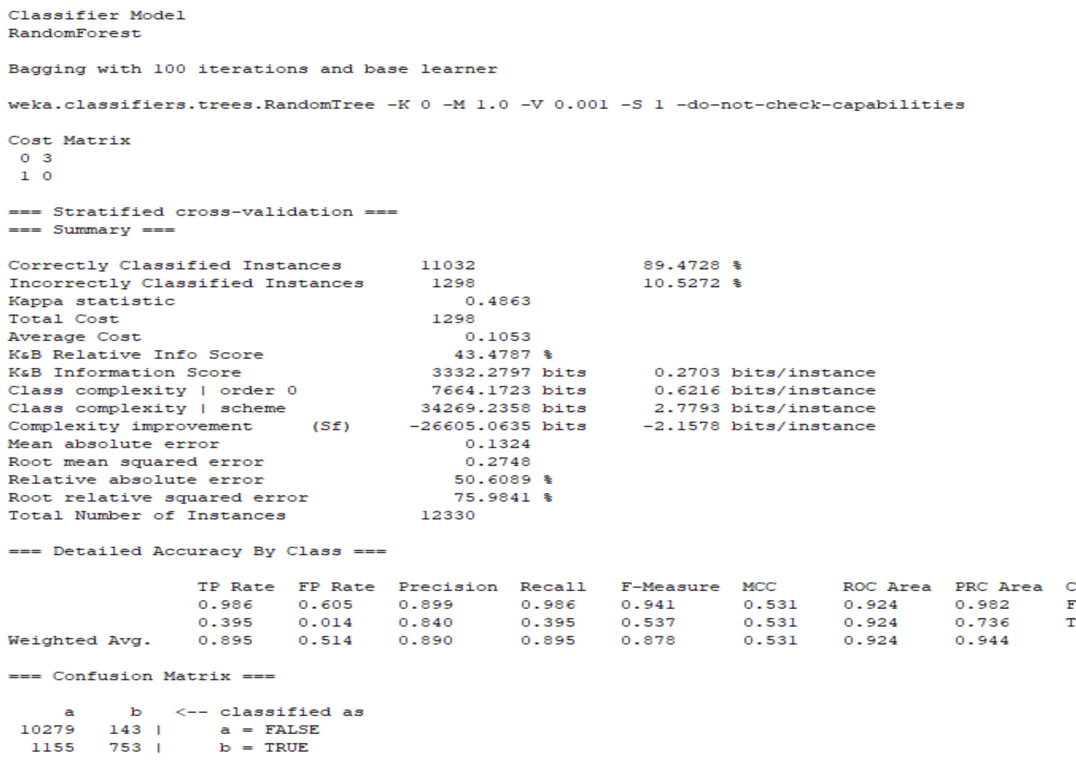
- **<u>Models tested in section 8.6</u>**

Cost-sensitive Random forest (1,1):



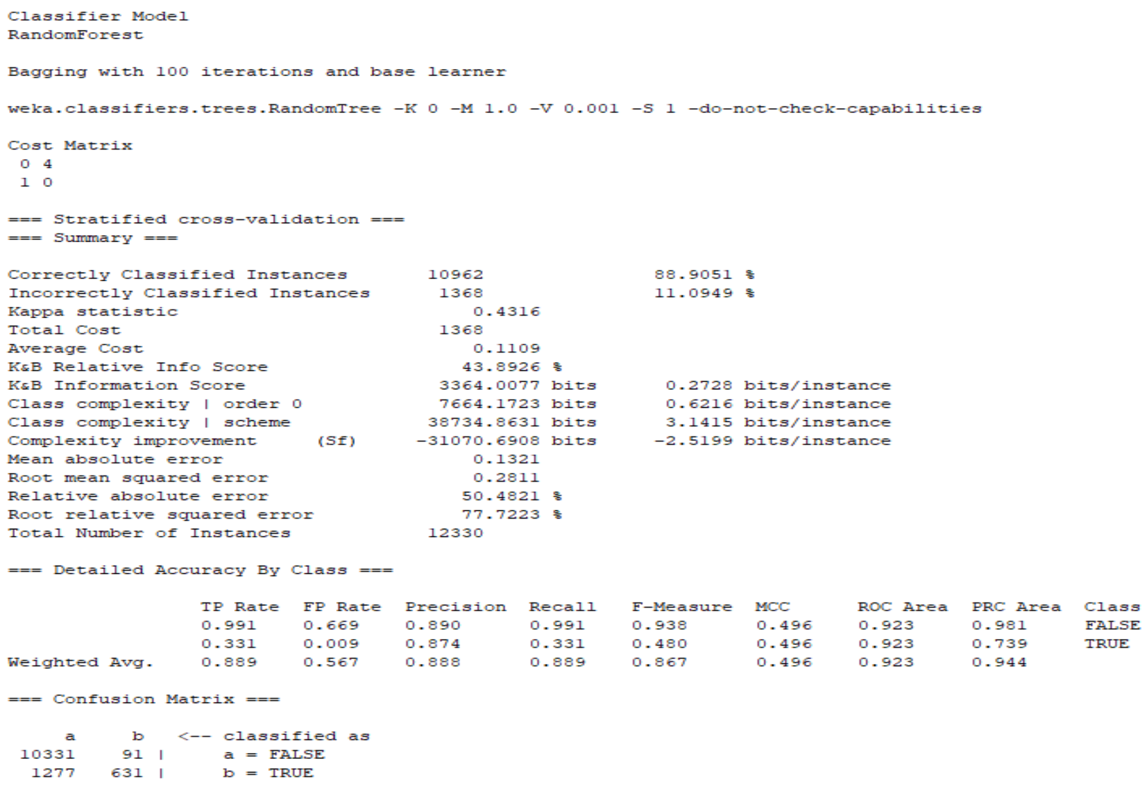
```
Classifier Model
RandomForest

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

Cost Matrix
 0 1
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11116               90.1541 %
Incorrectly Classified Instances      1214                9.8459 %
Kappa statistic                          0.5912
Total Cost                            1214
Average Cost                             0.0985
K&B Relative Info Score                 37.7517 %
K&B Information Score                 2893.3556 bits      0.2347 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            11524.2368 bits      0.9347 bits/instance
Complexity improvement     (Sf)      -3860.0645 bits     -0.3131 bits/instance
Mean absolute error                      0.1408
Root mean squared error                  0.2652
Relative absolute error                 53.8212 %
Root relative squared error             73.336  %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.960    0.415    0.927      0.960    0.943      0.596    0.929     0.985     FALSE
                 0.585    0.040    0.726      0.585    0.648      0.596    0.929     0.738     TRUE
Weighted Avg.    0.902    0.357    0.896      0.902    0.897      0.596    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10000   422 |     a = FALSE
   792  1116 |     b = TRUE
```

Cost-sensitive Random forest (1,3):



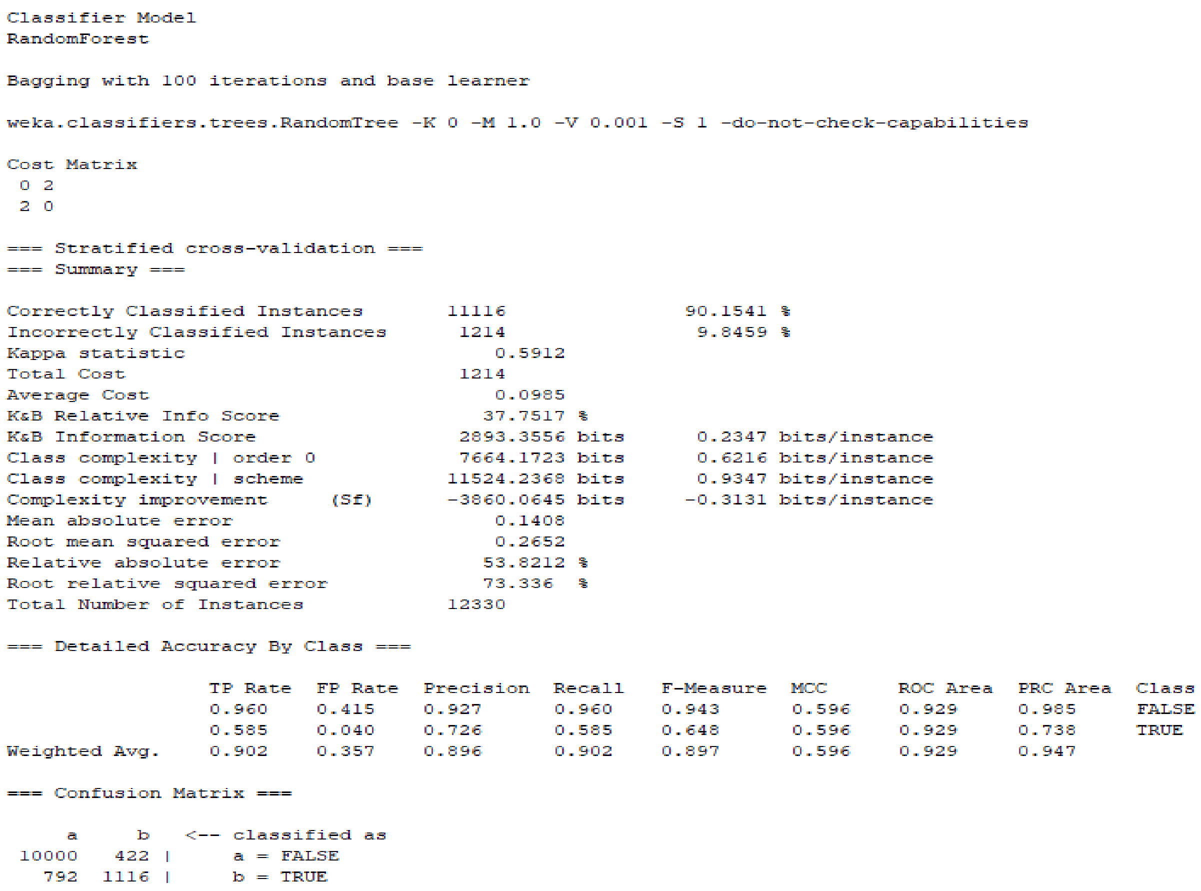
```
Classifier Model
RandomForest

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

Cost Matrix
 0 3
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11032               89.4728 %
Incorrectly Classified Instances      1298               10.5272 %
Kappa statistic                          0.4863
Total Cost                            1298
Average Cost                             0.1053
K&B Relative Info Score                 43.4787 %
K&B Information Score                 3332.2797 bits      0.2703 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            34269.2358 bits      2.7793 bits/instance
Complexity improvement     (Sf)     -26605.0635 bits     -2.1578 bits/instance
Mean absolute error                      0.1324
Root mean squared error                  0.2748
Relative absolute error                 50.6089 %
Root relative squared error             75.9841 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.986    0.605    0.899      0.986    0.941      0.531    0.924     0.982     FALSE
                 0.395    0.014    0.840      0.395    0.537      0.531    0.924     0.736     TRUE
Weighted Avg.    0.895    0.514    0.890      0.895    0.878      0.531    0.924     0.944

=== Confusion Matrix ===

     a     b   <-- classified as
 10279   143 |     a = FALSE
  1155   753 |     b = TRUE
```

Cost-sensitive Random forest (1,4):



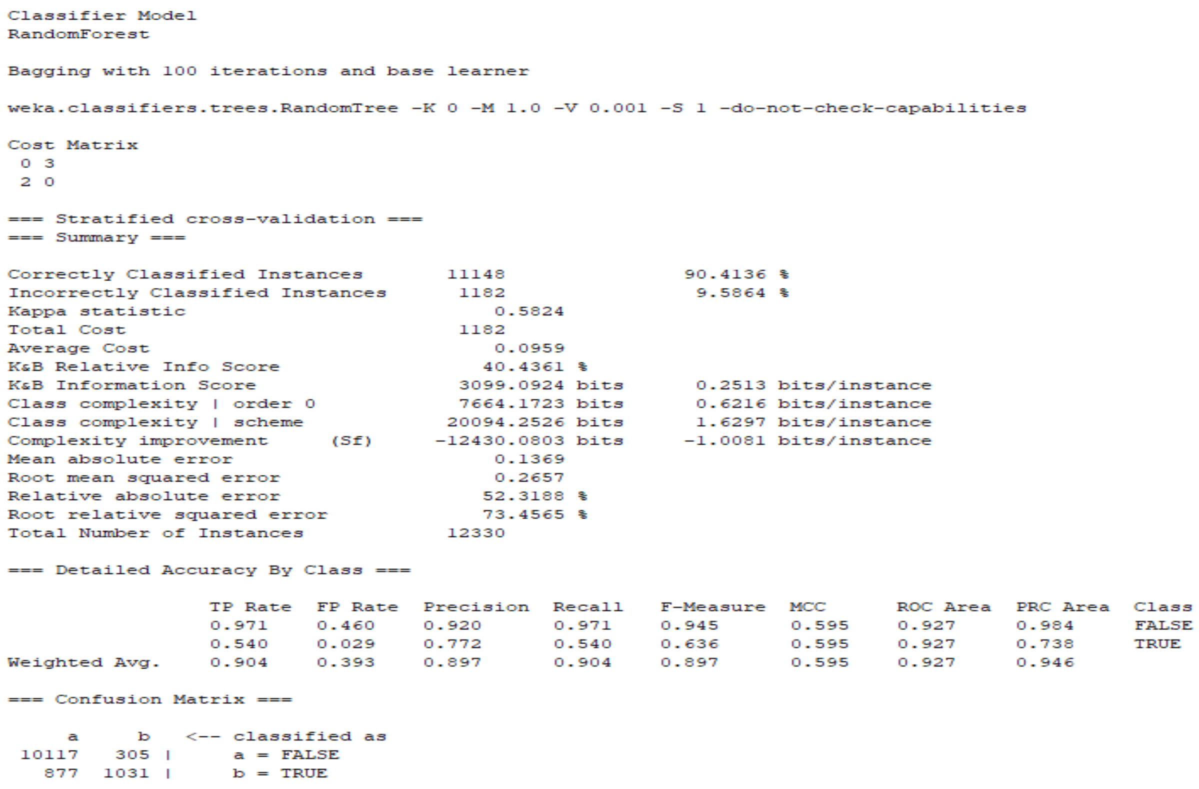
```
Classifier Model
RandomForest

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

Cost Matrix
 0 4
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10962               88.9051 %
Incorrectly Classified Instances      1368               11.0949 %
Kappa statistic                          0.4316
Total Cost                            1368
Average Cost                             0.1109
K&B Relative Info Score                 43.8926 %
K&B Information Score                 3364.0077 bits      0.2728 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            38734.8631 bits      3.1415 bits/instance
Complexity improvement     (Sf)     -31070.6908 bits     -2.5199 bits/instance
Mean absolute error                      0.1321
Root mean squared error                  0.2811
Relative absolute error                 50.4821 %
Root relative squared error             77.7223 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.991    0.669    0.890      0.991    0.938      0.496    0.923     0.981     FALSE
                 0.331    0.009    0.874      0.331    0.480      0.496    0.923     0.739     TRUE
Weighted Avg.    0.889    0.567    0.888      0.889    0.867      0.496    0.923     0.944

=== Confusion Matrix ===

     a     b   <-- classified as
 10331    91 |     a = FALSE
  1277   631 |     b = TRUE
```

Cost-sensitive Random forest (2,2):



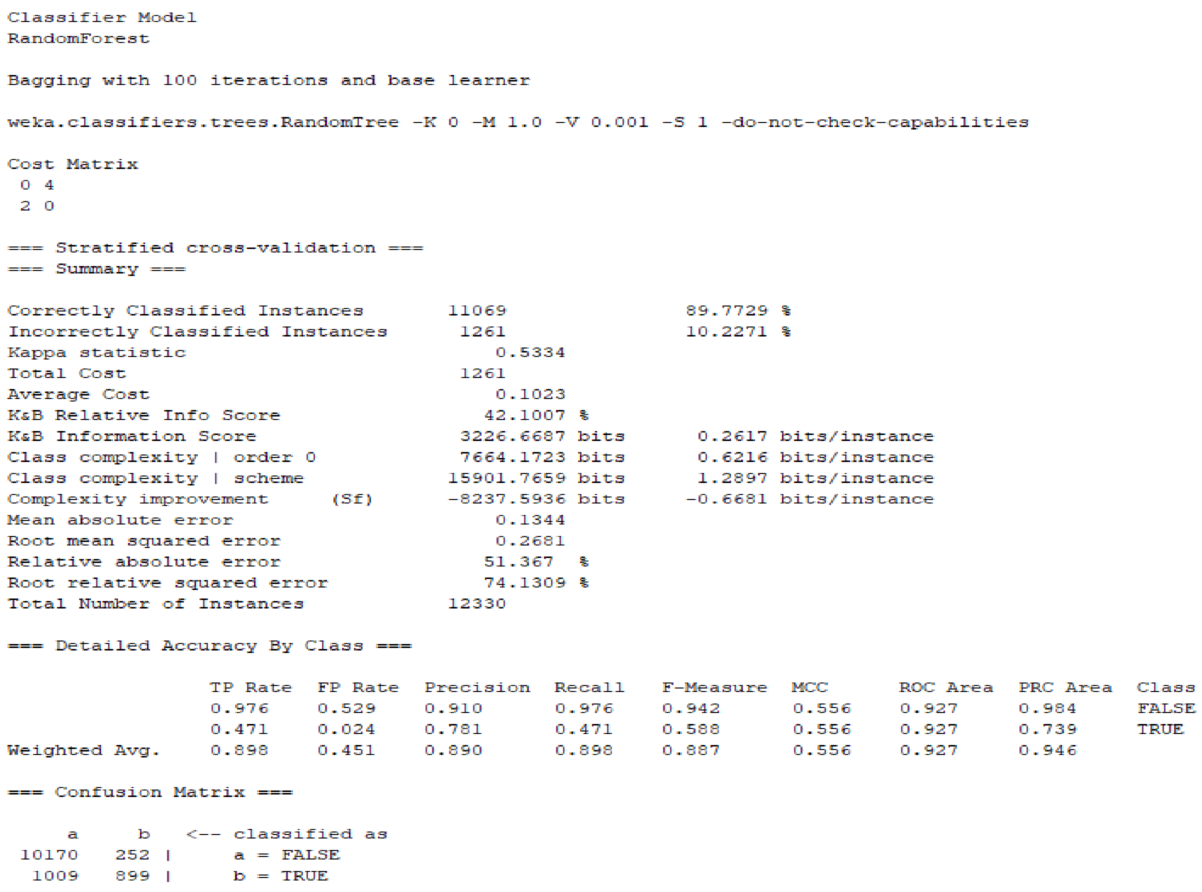
```
Classifier Model
RandomForest

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

Cost Matrix
 0 2
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11116               90.1541 %
Incorrectly Classified Instances      1214                9.8459 %
Kappa statistic                          0.5912
Total Cost                            1214
Average Cost                             0.0985
K&B Relative Info Score                 37.7517 %
K&B Information Score                 2893.3556 bits      0.2347 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            11524.2368 bits      0.9347 bits/instance
Complexity improvement     (Sf)      -3860.0645 bits     -0.3131 bits/instance
Mean absolute error                      0.1408
Root mean squared error                  0.2652
Relative absolute error                 53.8212 %
Root relative squared error             73.336  %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.960    0.415    0.927      0.960    0.943      0.596    0.929     0.985     FALSE
                 0.585    0.040    0.726      0.585    0.648      0.596    0.929     0.738     TRUE
Weighted Avg.    0.902    0.357    0.896      0.902    0.897      0.596    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10000   422 |     a = FALSE
   792  1116 |     b = TRUE
```

Cost-sensitive Random forest (2,3):



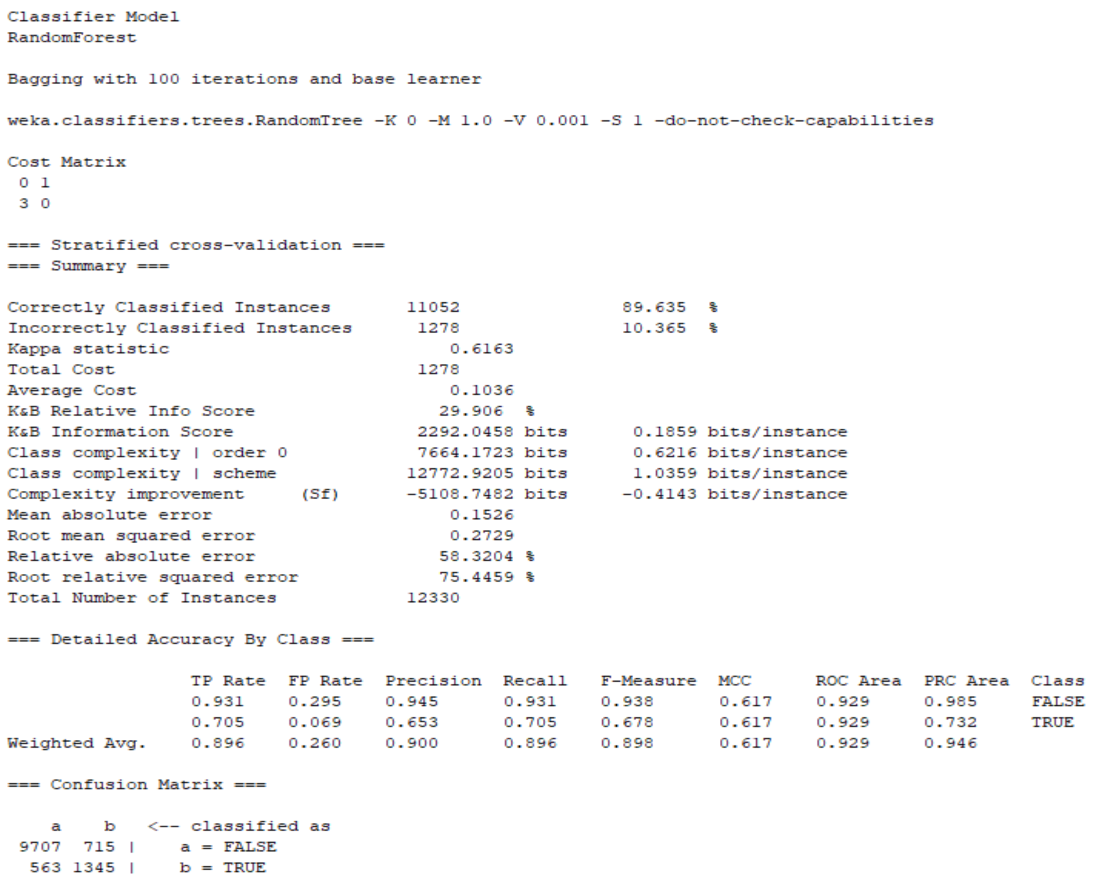
```
Classifier Model
RandomForest

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

Cost Matrix
 0 3
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11148               90.4136 %
Incorrectly Classified Instances      1182                9.5864 %
Kappa statistic                          0.5824
Total Cost                            1182
Average Cost                             0.0959
K&B Relative Info Score                 40.4361 %
K&B Information Score                 3099.0924 bits      0.2513 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            20094.2526 bits      1.6297 bits/instance
Complexity improvement     (Sf)     -12430.0803 bits     -1.0081 bits/instance
Mean absolute error                      0.1369
Root mean squared error                  0.2657
Relative absolute error                 52.3188 %
Root relative squared error             73.4565 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.971    0.460    0.920      0.971    0.945      0.595    0.927     0.984     FALSE
                 0.540    0.029    0.772      0.540    0.636      0.595    0.927     0.738     TRUE
Weighted Avg.    0.904    0.393    0.897      0.904    0.897      0.595    0.927     0.946

=== Confusion Matrix ===

     a     b   <-- classified as
 10117   305 |     a = FALSE
   877  1031 |     b = TRUE
```

Cost-sensitive Random forest (2,4):



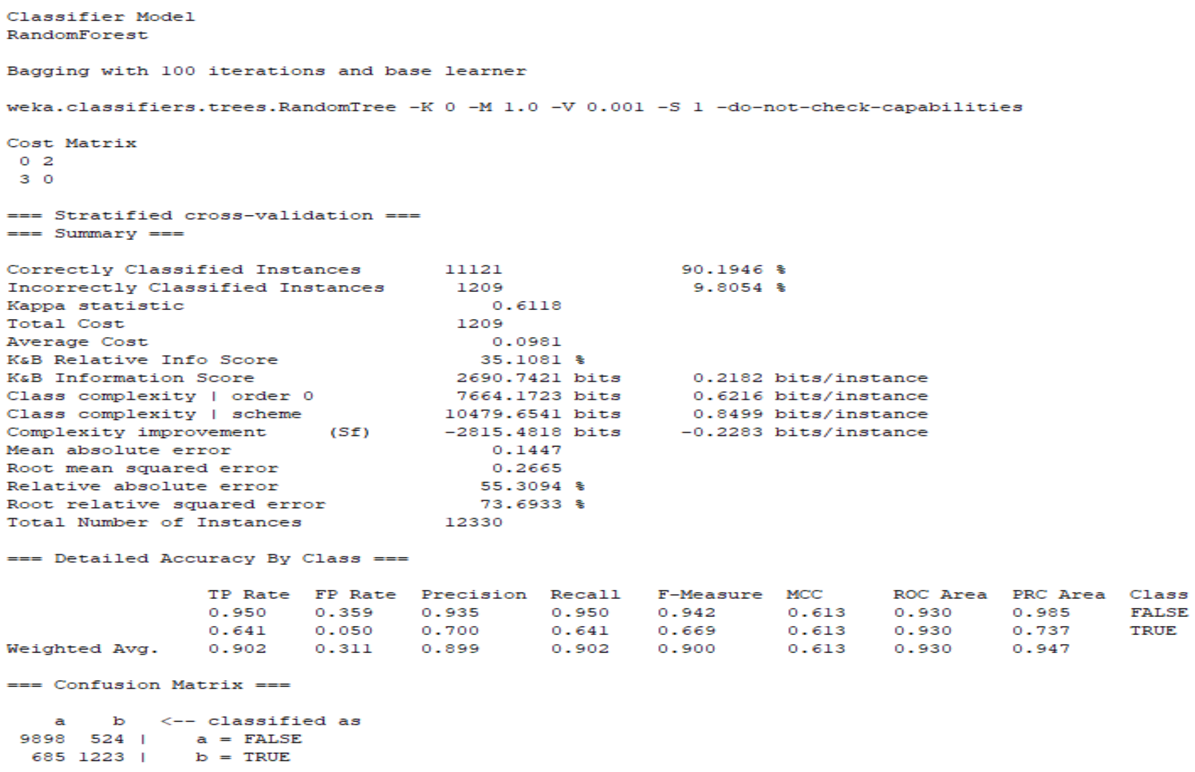
```
Classifier Model
RandomForest

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

Cost Matrix
 0 4
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11069               89.7729 %
Incorrectly Classified Instances      1261               10.2271 %
Kappa statistic                          0.5334
Total Cost                            1261
Average Cost                             0.1023
K&B Relative Info Score                 42.1007 %
K&B Information Score                 3226.6687 bits      0.2617 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            15901.7659 bits      1.2897 bits/instance
Complexity improvement     (Sf)      -8237.5936 bits     -0.6681 bits/instance
Mean absolute error                      0.1344
Root mean squared error                  0.2681
Relative absolute error                 51.367  %
Root relative squared error             74.1309 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.976    0.529    0.910      0.976    0.942      0.556    0.927     0.984     FALSE
                 0.471    0.024    0.781      0.471    0.588      0.556    0.927     0.739     TRUE
Weighted Avg.    0.898    0.451    0.890      0.898    0.887      0.556    0.927     0.946

=== Confusion Matrix ===

     a     b   <-- classified as
 10170   252 |     a = FALSE
  1009   899 |     b = TRUE
```

Cost-sensitive Random forest (3,1):



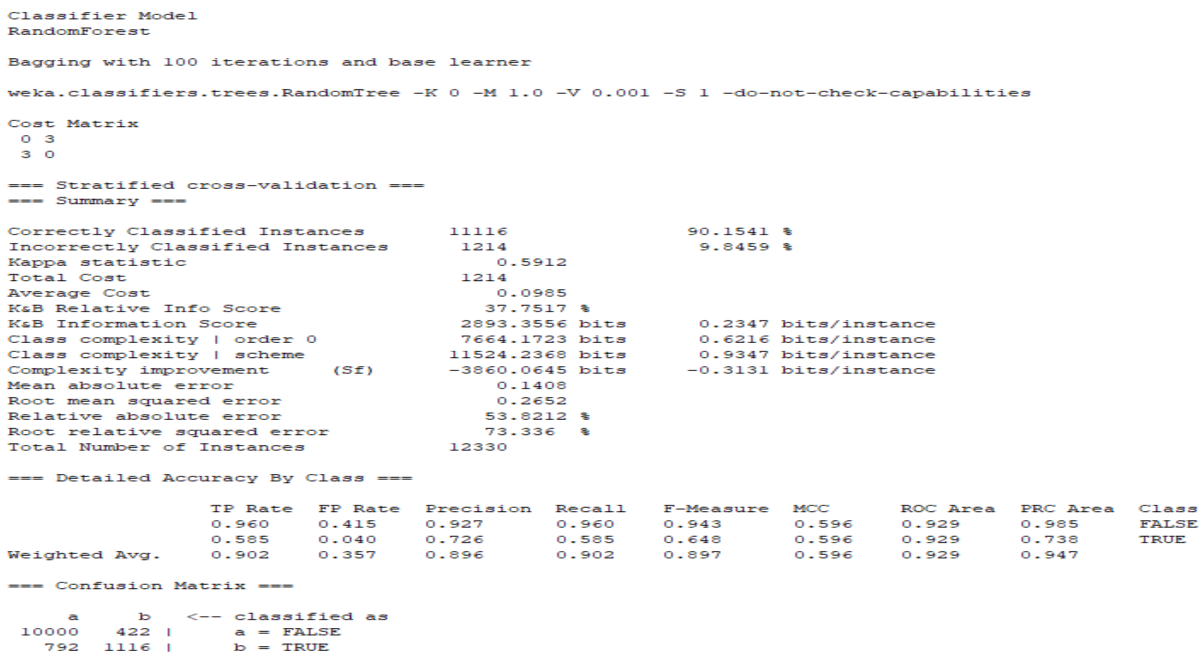
```
Classifier Model
RandomForest

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

Cost Matrix
 0 1
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11052               89.635  %
Incorrectly Classified Instances      1278               10.365  %
Kappa statistic                          0.6163
Total Cost                            1278
Average Cost                             0.1036
K&B Relative Info Score                 29.906  %
K&B Information Score                 2292.0458 bits      0.1859 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            12772.9205 bits      1.0359 bits/instance
Complexity improvement     (Sf)      -5108.7482 bits     -0.4143 bits/instance
Mean absolute error                      0.1526
Root mean squared error                  0.2729
Relative absolute error                 58.3204 %
Root relative squared error             75.4459 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.931    0.295    0.945      0.931    0.938      0.617    0.929     0.985     FALSE
                 0.705    0.069    0.653      0.705    0.678      0.617    0.929     0.732     TRUE
Weighted Avg.    0.896    0.260    0.900      0.896    0.898      0.617    0.929     0.946

=== Confusion Matrix ===

    a    b   <-- classified as
 9707  715 |    a = FALSE
  563 1345 |    b = TRUE
```

Cost-sensitive Random forest (3,2):



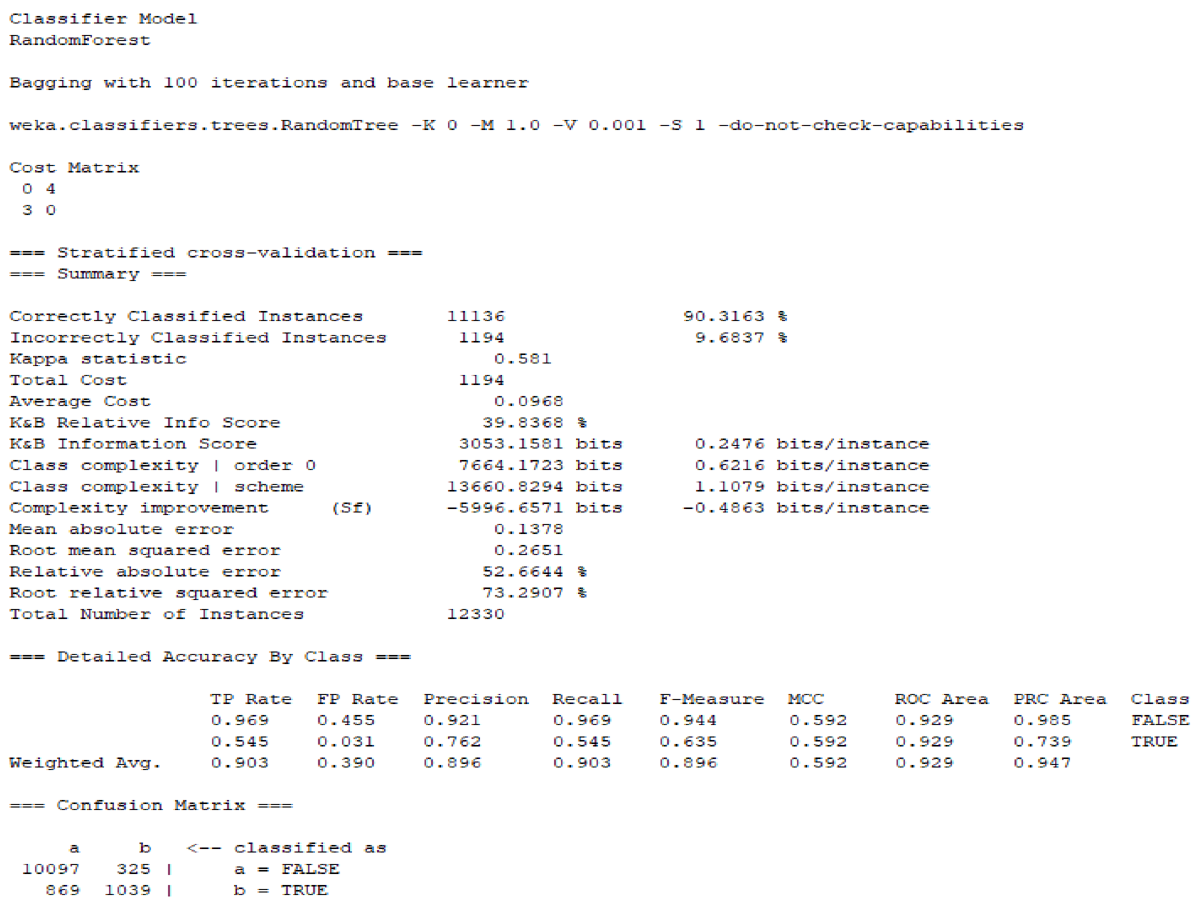
```
Classifier Model
RandomForest

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

Cost Matrix
 0 2
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11121               90.1946 %
Incorrectly Classified Instances      1209                9.8054 %
Kappa statistic                          0.6118
Total Cost                            1209
Average Cost                             0.0981
K&B Relative Info Score                 35.1081 %
K&B Information Score                 2690.7421 bits      0.2182 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            10479.6541 bits      0.8499 bits/instance
Complexity improvement     (Sf)      -2815.4818 bits     -0.2283 bits/instance
Mean absolute error                      0.1447
Root mean squared error                  0.2665
Relative absolute error                 55.3094 %
Root relative squared error             73.6933 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.950    0.359    0.935      0.950    0.942      0.613    0.930     0.985     FALSE
                 0.641    0.050    0.700      0.641    0.669      0.613    0.930     0.737     TRUE
Weighted Avg.    0.902    0.311    0.899      0.902    0.900      0.613    0.930     0.947

=== Confusion Matrix ===

    a    b   <-- classified as
 9898  524 |    a = FALSE
  685 1223 |    b = TRUE
```

Cost-sensitive Random forest (3,3):



```
Classifier Model
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

Cost Matrix
 0 3
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11116               90.1541 %
Incorrectly Classified Instances      1214                9.8459 %
Kappa statistic                          0.5912
Total Cost                            1214
Average Cost                             0.0985
K&B Relative Info Score                 37.7517 %
K&B Information Score                 2893.3556 bits      0.2347 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            11524.2368 bits      0.9347 bits/instance
Complexity improvement     (Sf)      -3860.0645 bits     -0.3131 bits/instance
Mean absolute error                      0.1408
Root mean squared error                  0.2652
Relative absolute error                 53.8212 %
Root relative squared error             73.336  %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.960    0.415    0.927      0.960    0.943      0.596    0.929     0.985     FALSE
                 0.585    0.040    0.726      0.585    0.648      0.596    0.929     0.738     TRUE
Weighted Avg.    0.902    0.357    0.896      0.902    0.897      0.596    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10000   422 |     a = FALSE
   792  1116 |     b = TRUE
```

Cost-sensitive Random forest (3,4):



```
Classifier Model
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

Cost Matrix
 0 4
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11136               90.3163 %
Incorrectly Classified Instances      1194                9.6837 %
Kappa statistic                          0.581
Total Cost                            1194
Average Cost                             0.0968
K&B Relative Info Score                 39.8368 %
K&B Information Score                 3053.1581 bits      0.2476 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            13660.8294 bits      1.1079 bits/instance
Complexity improvement     (Sf)      -5996.6571 bits     -0.4863 bits/instance
Mean absolute error                      0.1378
Root mean squared error                  0.2651
Relative absolute error                 52.6644 %
Root relative squared error             73.2907 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.969    0.455    0.921      0.969    0.944      0.592    0.929     0.985     FALSE
                 0.545    0.031    0.762      0.545    0.635      0.592    0.929     0.739     TRUE
Weighted Avg.    0.903    0.390    0.896      0.903    0.896      0.592    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10097   325 |     a = FALSE
   869  1039 |     b = TRUE
```

Cost-sensitive Random forest (4,1):

```
Classifier Model
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

Cost Matrix
 0 1
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11024               89.4079 %
Incorrectly Classified Instances      1306               10.5921 %
Kappa statistic                          0.6133
Total Cost                            1306
Average Cost                             0.1059
K&B Relative Info Score                 27.587  %
K&B Information Score                 2114.3129 bits      0.1715 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             7500.5436 bits      0.6083 bits/instance
Complexity improvement     (Sf)        163.6287 bits      0.0133 bits/instance
Mean absolute error                      0.1563
Root mean squared error                  0.2755
Relative absolute error                 59.7357 %
Root relative squared error             76.1683 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.927    0.285    0.947      0.927    0.937      0.615    0.930     0.986     FALSE
                 0.715    0.073    0.641      0.715    0.676      0.615    0.930     0.734     TRUE
Weighted Avg.    0.894    0.252    0.900      0.894    0.896      0.615    0.930     0.947

=== Confusion Matrix ===

    a    b   <-- classified as
 9659  763 |    a = FALSE
  543 1365 |    b = TRUE
```

Cost-sensitive Random forest (4,2):

```
Classifier Model
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

Cost Matrix
 0 2
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11114               90.1379 %
Incorrectly Classified Instances      1216                9.8621 %
Kappa statistic                          0.6194
Total Cost                            1216
Average Cost                             0.0986
K&B Relative Info Score                 32.9701 %
K&B Information Score                 2526.8869 bits      0.2049 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            11608.5301 bits      0.9415 bits/instance
Complexity improvement     (Sf)      -3944.3578 bits     -0.3199 bits/instance
Mean absolute error                      0.148
Root mean squared error                  0.2685
Relative absolute error                 56.5591 %
Root relative squared error             74.2526 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.944    0.330    0.940      0.944    0.942      0.619    0.929     0.985     FALSE
                 0.670    0.056    0.686      0.670    0.678      0.619    0.929     0.737     TRUE
Weighted Avg.    0.901    0.288    0.900      0.901    0.901      0.619    0.929     0.947

=== Confusion Matrix ===

    a    b   <-- classified as
 9836  586 |    a = FALSE
  630 1278 |    b = TRUE
```

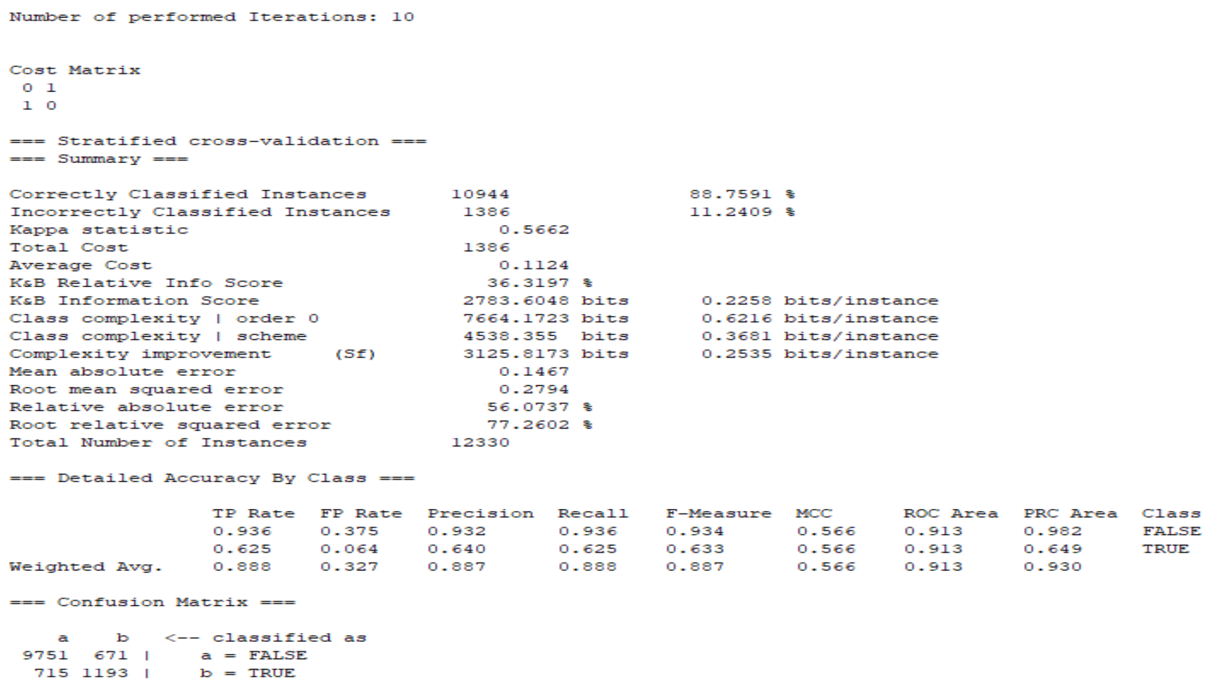
Cost-sensitive Random forest (4,3):

```
Classifier Model
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

Cost Matrix
 0 3
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11110               90.1054 %
Incorrectly Classified Instances      1220                9.8946 %
Kappa statistic                          0.6038
Total Cost                            1220
Average Cost                             0.0989
K&B Relative Info Score                 35.8885 %
K&B Information Score                 2750.556  bits      0.2231 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             9406.6117 bits      0.7629 bits/instance
Complexity improvement     (Sf)      -1742.4394 bits     -0.1413 bits/instance
Mean absolute error                      0.1437
Root mean squared error                  0.2663
Relative absolute error                 54.9114 %
Root relative squared error             73.6282 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.952    0.375    0.933      0.952    0.942      0.605    0.929     0.985     FALSE
                 0.625    0.048    0.703      0.625    0.661      0.605    0.929     0.737     TRUE
Weighted Avg.    0.901    0.325    0.897      0.901    0.899      0.605    0.929     0.947

=== Confusion Matrix ===

    a    b   <-- classified as
 9918  504 |    a = FALSE
  716 1192 |    b = TRUE
```

Cost-sensitive Random forest (4,4):

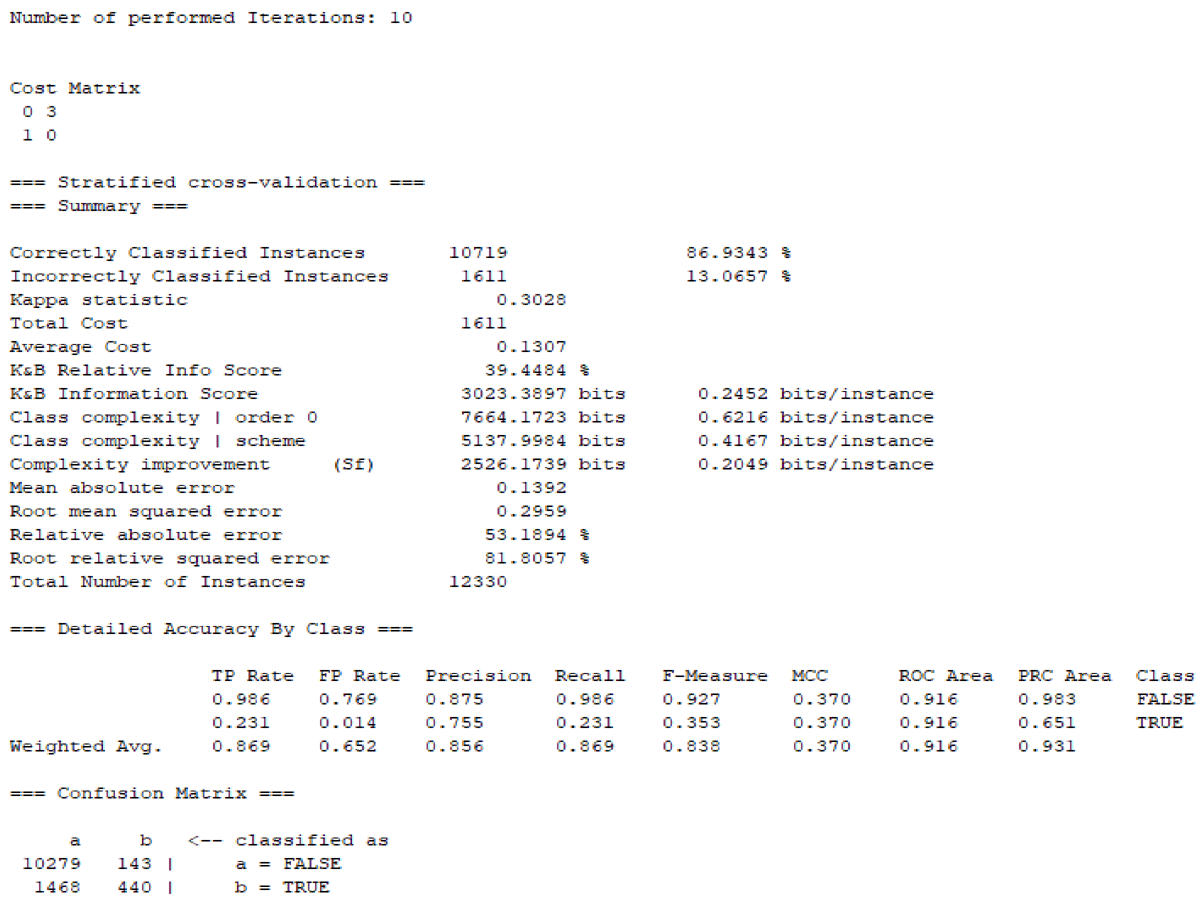
```
Classifier Model
RandomForest

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

Cost Matrix
 0 4
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11116               90.1541 %
Incorrectly Classified Instances      1214                9.8459 %
Kappa statistic                          0.5912
Total Cost                            1214
Average Cost                             0.0985
K&B Relative Info Score                 37.7517 %
K&B Information Score                 2893.3556 bits      0.2347 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme            11524.2368 bits      0.9347 bits/instance
Complexity improvement     (Sf)      -3860.0645 bits     -0.3131 bits/instance
Mean absolute error                      0.1408
Root mean squared error                  0.2652
Relative absolute error                 53.8212 %
Root relative squared error             73.336  %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.960    0.415    0.927      0.960    0.943      0.596    0.929     0.985     FALSE
                 0.585    0.040    0.726      0.585    0.648      0.596    0.929     0.738     TRUE
Weighted Avg.    0.902    0.357    0.896      0.902    0.897      0.596    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10000   422 |     a = FALSE
   792  1116 |     b = TRUE
```

Boosting (AddBoostM1 Cost) (1,1)



```
Number of performed Iterations: 10


Cost Matrix
 0 1
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10944               88.7591 %
Incorrectly Classified Instances      1386               11.2409 %
Kappa statistic                          0.5662
Total Cost                            1386
Average Cost                             0.1124
K&B Relative Info Score                 36.3197 %
K&B Information Score                 2783.6048 bits      0.2258 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4538.355  bits      0.3681 bits/instance
Complexity improvement     (Sf)       3125.8173 bits      0.2535 bits/instance
Mean absolute error                      0.1467
Root mean squared error                  0.2794
Relative absolute error                 56.0737 %
Root relative squared error             77.2602 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.936    0.375    0.932      0.936    0.934      0.566    0.913     0.982     FALSE
                 0.625    0.064    0.640      0.625    0.633      0.566    0.913     0.649     TRUE
Weighted Avg.    0.888    0.327    0.887      0.888    0.887      0.566    0.913     0.930

=== Confusion Matrix ===

    a    b   <-- classified as
 9751  671 |    a = FALSE
  715 1193 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (1,3)



```
Number of performed Iterations: 10


Cost Matrix
 0 3
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10719               86.9343 %
Incorrectly Classified Instances      1611               13.0657 %
Kappa statistic                          0.3028
Total Cost                            1611
Average Cost                             0.1307
K&B Relative Info Score                 39.4484 %
K&B Information Score                 3023.3897 bits      0.2452 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             5137.9984 bits      0.4167 bits/instance
Complexity improvement     (Sf)       2526.1739 bits      0.2049 bits/instance
Mean absolute error                      0.1392
Root mean squared error                  0.2959
Relative absolute error                 53.1894 %
Root relative squared error             81.8057 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.986    0.769    0.875      0.986    0.927      0.370    0.916     0.983     FALSE
                 0.231    0.014    0.755      0.231    0.353      0.370    0.916     0.651     TRUE
Weighted Avg.    0.869    0.652    0.856      0.869    0.838      0.370    0.916     0.931

=== Confusion Matrix ===

     a     b   <-- classified as
 10279   143 |     a = FALSE
  1468   440 |     b = TRUE
```

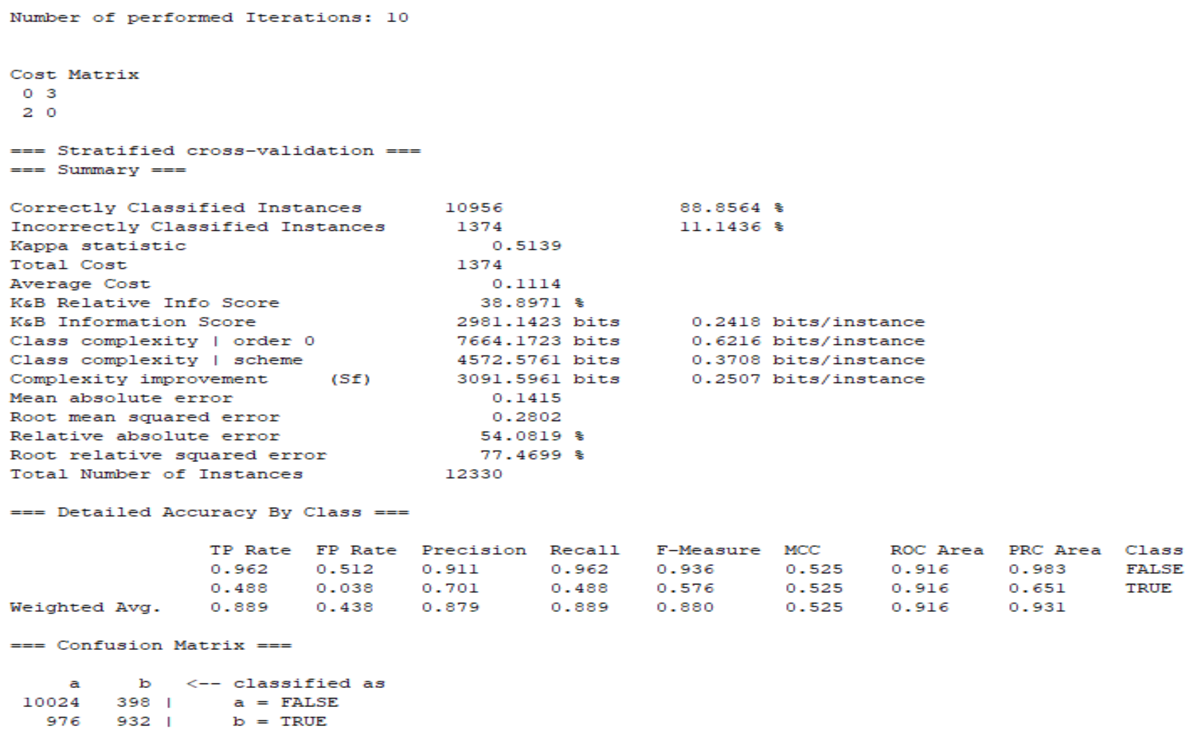
Boosting (AddBoostM1 Cost) (1,4)

```
Number of performed Iterations: 10


Cost Matrix
 0 4
 1 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10695               86.7397 %
Incorrectly Classified Instances      1635               13.2603 %
Kappa statistic                          0.2786
Total Cost                            1635
Average Cost                             0.1326
K&B Relative Info Score                 38.7904 %
K&B Information Score                 2972.9595 bits      0.2411 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             5507.9016 bits      0.4467 bits/instance
Complexity improvement     (Sf)       2156.2707 bits      0.1749 bits/instance
Mean absolute error                      0.1398
Root mean squared error                  0.3053
Relative absolute error                 53.4159 %
Root relative squared error             84.4047 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.988    0.792    0.872      0.988    0.926      0.353    0.916     0.983     FALSE
                 0.208    0.012    0.763      0.208    0.326      0.353    0.916     0.651     TRUE
Weighted Avg.    0.867    0.672    0.855      0.867    0.834      0.353    0.916     0.931

=== Confusion Matrix ===

     a     b   <-- classified as
 10299   123 |     a = FALSE
  1512   396 |     b = TRUE
```

Boosting (AddBoostM1 Cost) (2,2)

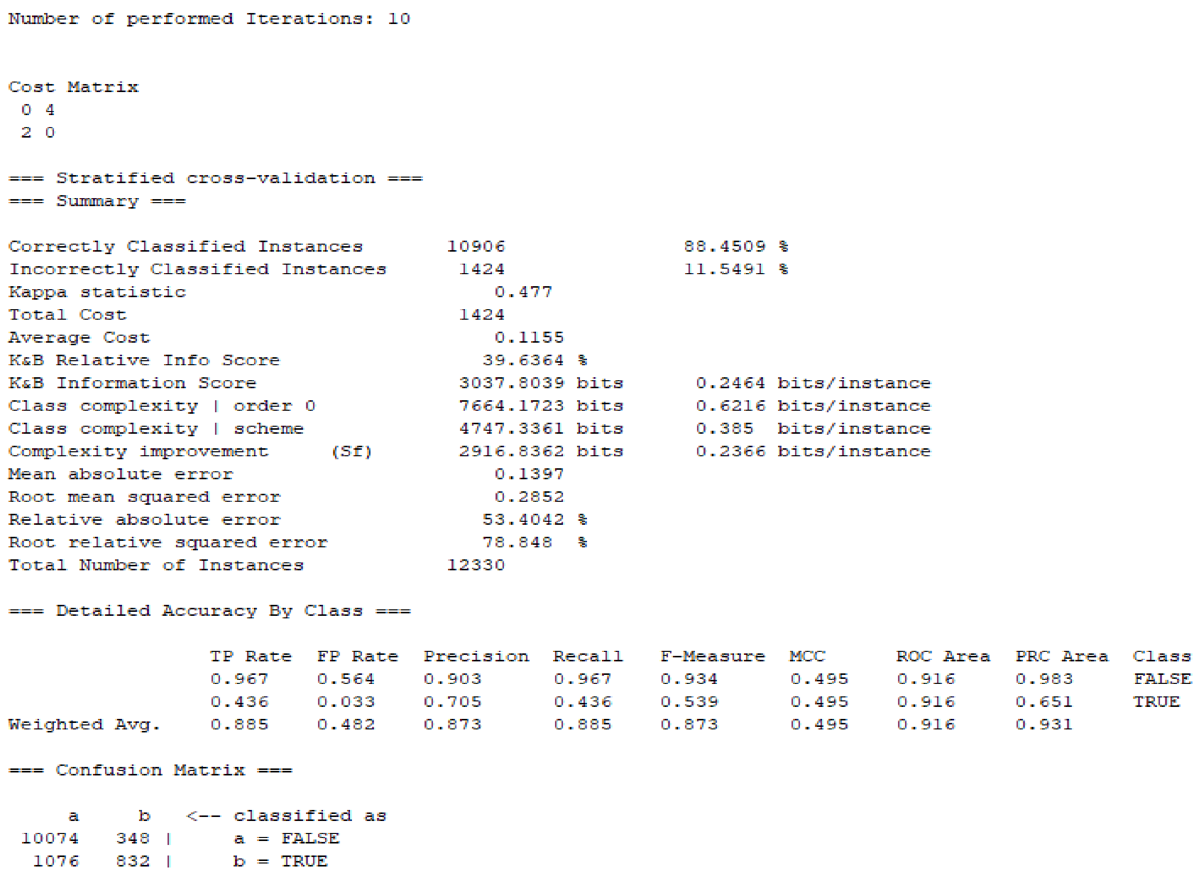
```
Number of performed Iterations: 10


Cost Matrix
 0 2
 2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10944               88.7591 %
Incorrectly Classified Instances      1386               11.2409 %
Kappa statistic                          0.5662
Total Cost                            1386
Average Cost                             0.1124
K&B Relative Info Score                 36.3197 %
K&B Information Score                 2783.6048 bits      0.2258 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4538.355  bits      0.3681 bits/instance
Complexity improvement     (Sf)       3125.8173 bits      0.2535 bits/instance
Mean absolute error                      0.1467
Root mean squared error                  0.2794
Relative absolute error                 56.0737 %
Root relative squared error             77.2602 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.936    0.375    0.932      0.936    0.934      0.566    0.913     0.982     FALSE
                 0.625    0.064    0.640      0.625    0.633      0.566    0.913     0.649     TRUE
Weighted Avg.    0.888    0.327    0.887      0.888    0.887      0.566    0.913     0.930

=== Confusion Matrix ===

    a    b   <-- classified as
 9751  671 |    a = FALSE
  715 1193 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (2,3)



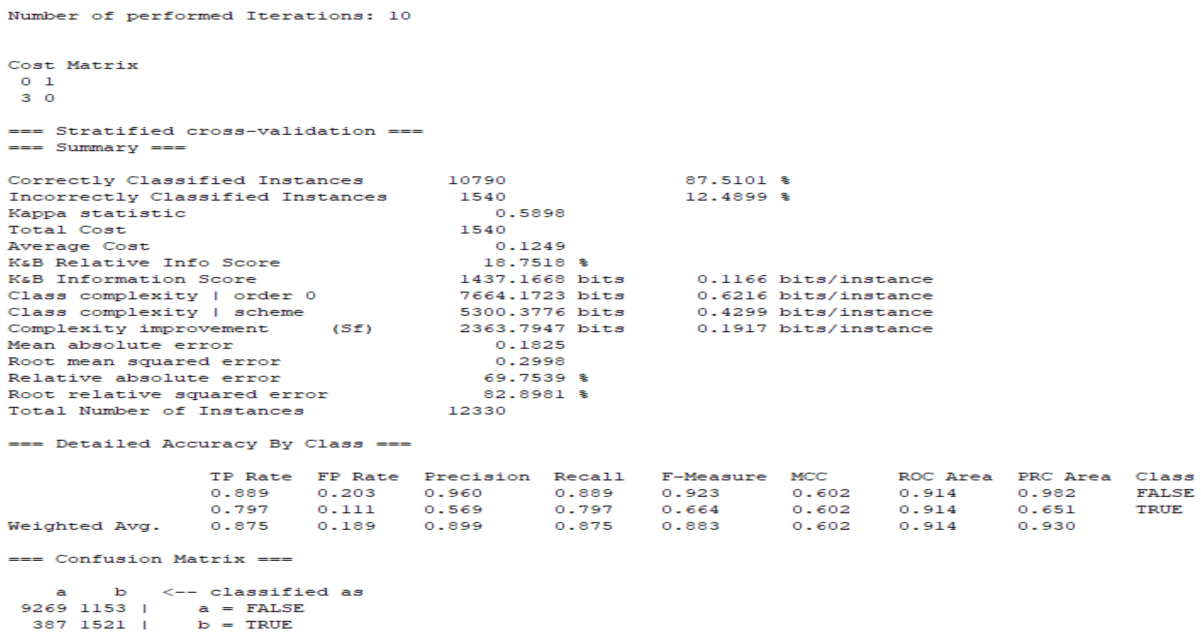
```
Number of performed Iterations: 10


Cost Matrix
  0 3
  2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10956               88.8564 %
Incorrectly Classified Instances      1374               11.1436 %
Kappa statistic                          0.5139
Total Cost                            1374
Average Cost                             0.1114
K&B Relative Info Score                 38.8971 %
K&B Information Score                 2981.1423 bits      0.2418 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4572.5761 bits      0.3708 bits/instance
Complexity improvement     (Sf)       3091.5961 bits      0.2507 bits/instance
Mean absolute error                      0.1415
Root mean squared error                  0.2802
Relative absolute error                 54.0819 %
Root relative squared error             77.4699 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.962    0.512    0.911      0.962    0.936      0.525    0.916     0.983     FALSE
                 0.488    0.038    0.701      0.488    0.576      0.525    0.916     0.651     TRUE
Weighted Avg.    0.889    0.438    0.879      0.889    0.880      0.525    0.916     0.931

=== Confusion Matrix ===

     a     b   <-- classified as
 10024   398 |     a = FALSE
   976   932 |     b = TRUE
```

Boosting (AddBoostM1 Cost) (2,4)



```
Number of performed Iterations: 10


Cost Matrix
  0 4
  2 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10906               88.4509 %
Incorrectly Classified Instances      1424               11.5491 %
Kappa statistic                          0.477
Total Cost                            1424
Average Cost                             0.1155
K&B Relative Info Score                 39.6364 %
K&B Information Score                 3037.8039 bits      0.2464 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4747.3361 bits      0.385  bits/instance
Complexity improvement     (Sf)       2916.8362 bits      0.2366 bits/instance
Mean absolute error                      0.1397
Root mean squared error                  0.2852
Relative absolute error                 53.4042 %
Root relative squared error             78.848  %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.967    0.564    0.903      0.967    0.934      0.495    0.916     0.983     FALSE
                 0.436    0.033    0.705      0.436    0.539      0.495    0.916     0.651     TRUE
Weighted Avg.    0.885    0.482    0.873      0.885    0.873      0.495    0.916     0.931

=== Confusion Matrix ===

     a     b   <-- classified as
 10074   348 |     a = FALSE
  1076   832 |     b = TRUE
```

Boosting (AddBoostM1 Cost) (3,1)

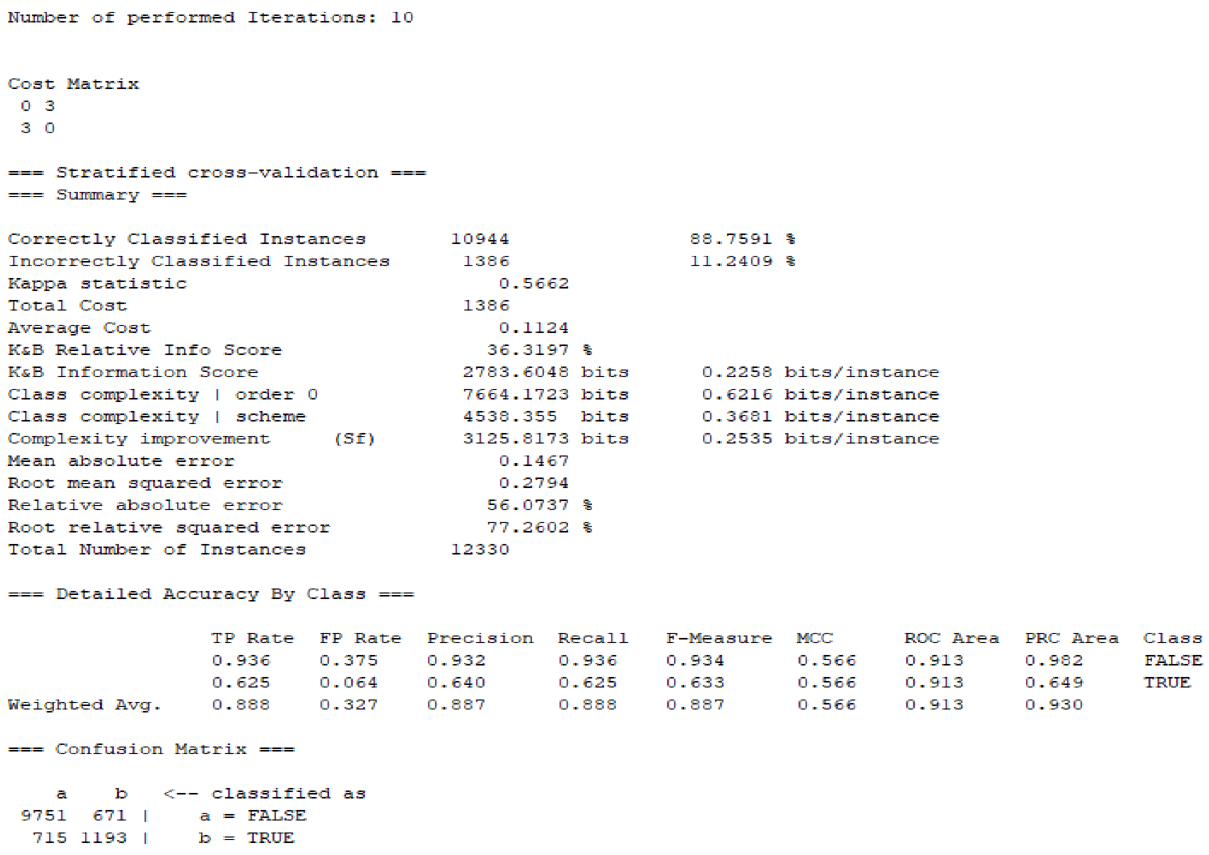


```
Number of performed Iterations: 10


Cost Matrix
 0 1
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10790               87.5101 %
Incorrectly Classified Instances      1540               12.4899 %
Kappa statistic                          0.5898
Total Cost                            1540
Average Cost                             0.1249
K&B Relative Info Score                 18.7518 %
K&B Information Score                 1437.1668 bits      0.1166 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             5300.3776 bits      0.4299 bits/instance
Complexity improvement     (Sf)       2363.7947 bits      0.1917 bits/instance
Mean absolute error                      0.1825
Root mean squared error                  0.2998
Relative absolute error                 69.7539 %
Root relative squared error             82.8981 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.889    0.203    0.960      0.889    0.923      0.602    0.914     0.982     FALSE
                 0.797    0.111    0.569      0.797    0.664      0.602    0.914     0.651     TRUE
Weighted Avg.    0.875    0.189    0.899      0.875    0.883      0.602    0.914     0.930

=== Confusion Matrix ===

    a    b   <-- classified as
 9269 1153 |    a = FALSE
  387 1521 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (3,2)

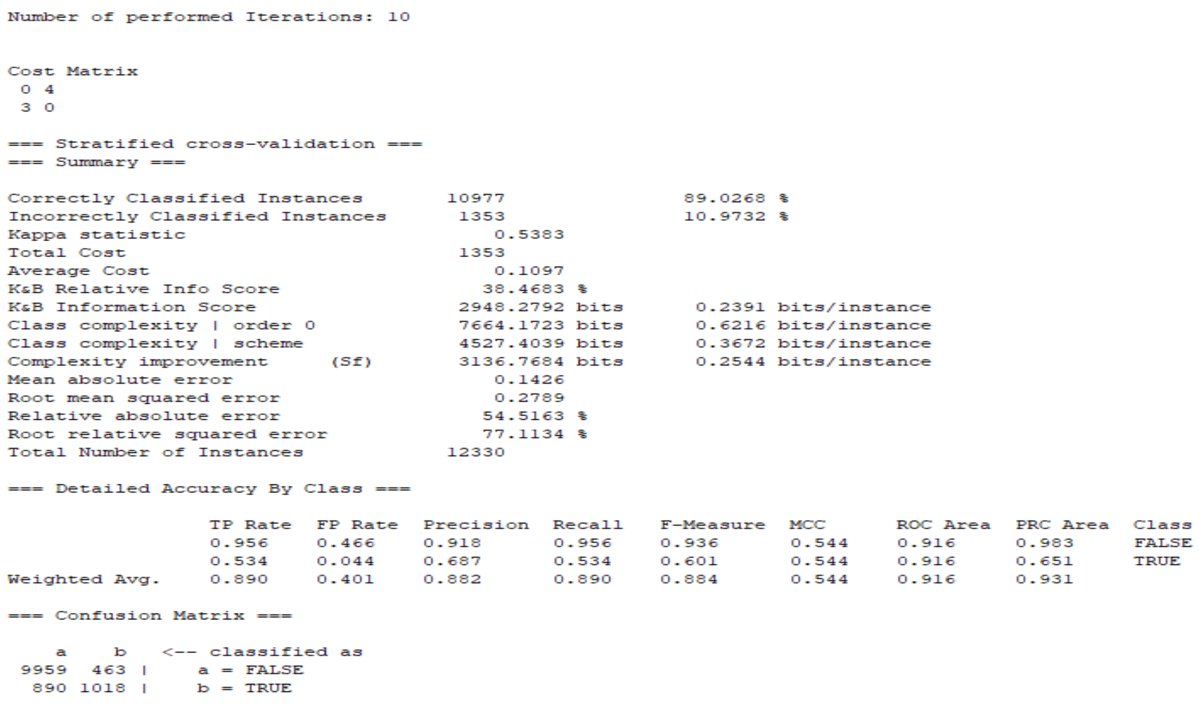
```
Number of performed Iterations: 10


Cost Matrix
 0 2
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10902               88.4185 %
Incorrectly Classified Instances      1428               11.5815 %
Kappa statistic                          0.5867
Total Cost                            1428
Average Cost                             0.1158
K&B Relative Info Score                 32.3049 %
K&B Information Score                 2475.904  bits      0.2008 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4669.0991 bits      0.3787 bits/instance
Complexity improvement     (Sf)       2995.0732 bits      0.2429 bits/instance
Mean absolute error                      0.1568
Root mean squared error                  0.2841
Relative absolute error                 59.9339 %
Root relative squared error             78.5604 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.916    0.287    0.946      0.916    0.930      0.589    0.915     0.982     FALSE
                 0.713    0.084    0.607      0.713    0.656      0.589    0.915     0.649     TRUE
Weighted Avg.    0.884    0.256    0.893      0.884    0.888      0.589    0.915     0.931

=== Confusion Matrix ===

    a    b   <-- classified as
 9542  880 |    a = FALSE
  548 1360 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (3,3)



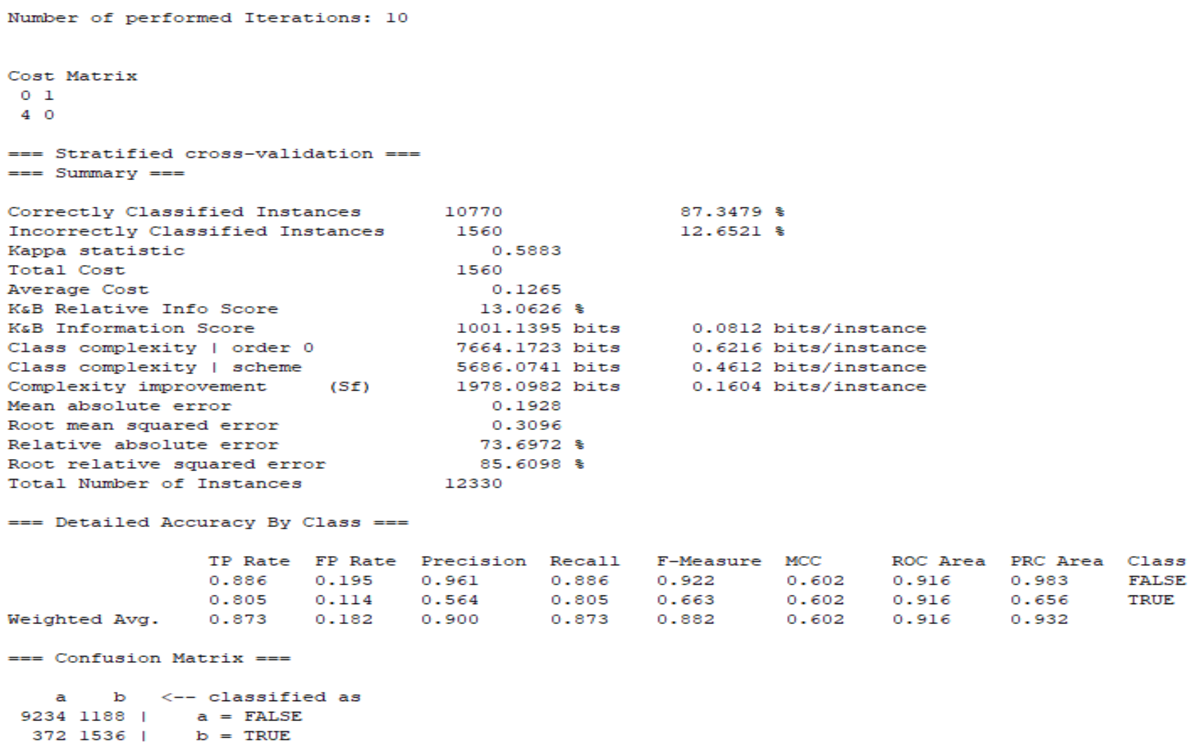
```
Number of performed Iterations: 10


Cost Matrix
 0 3
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10944               88.7591 %
Incorrectly Classified Instances      1386               11.2409 %
Kappa statistic                          0.5662
Total Cost                            1386
Average Cost                             0.1124
K&B Relative Info Score                 36.3197 %
K&B Information Score                 2783.6048 bits      0.2258 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4538.355  bits      0.3681 bits/instance
Complexity improvement     (Sf)       3125.8173 bits      0.2535 bits/instance
Mean absolute error                      0.1467
Root mean squared error                  0.2794
Relative absolute error                 56.0737 %
Root relative squared error             77.2602 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.936    0.375    0.932      0.936    0.934      0.566    0.913     0.982     FALSE
                 0.625    0.064    0.640      0.625    0.633      0.566    0.913     0.649     TRUE
Weighted Avg.    0.888    0.327    0.887      0.888    0.887      0.566    0.913     0.930

=== Confusion Matrix ===

    a    b   <-- classified as
 9751  671 |    a = FALSE
  715 1193 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (3,4)



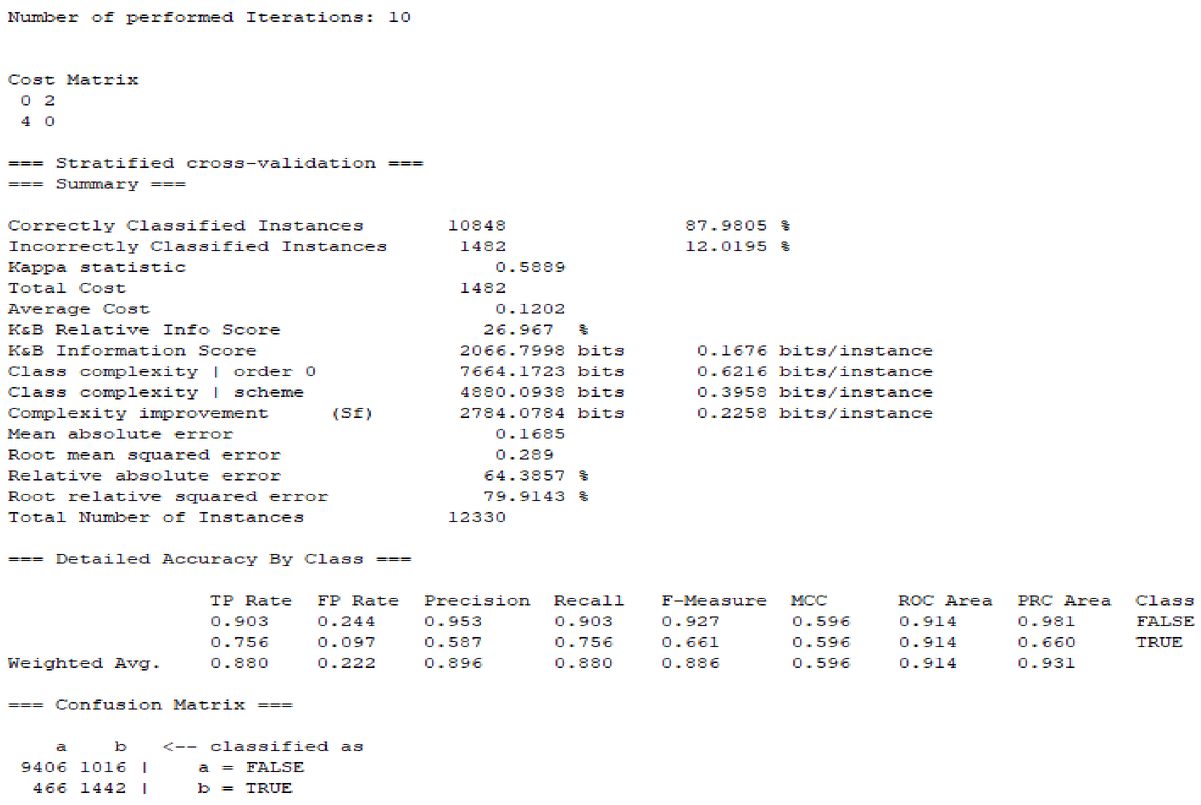
```
Number of performed Iterations: 10


Cost Matrix
 0 4
 3 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10977               89.0268 %
Incorrectly Classified Instances      1353               10.9732 %
Kappa statistic                          0.5383
Total Cost                            1353
Average Cost                             0.1097
K&B Relative Info Score                 38.4683 %
K&B Information Score                 2948.2792 bits      0.2391 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4527.4039 bits      0.3672 bits/instance
Complexity improvement     (Sf)       3136.7684 bits      0.2544 bits/instance
Mean absolute error                      0.1426
Root mean squared error                  0.2789
Relative absolute error                 54.5163 %
Root relative squared error             77.1134 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.956    0.466    0.918      0.956    0.936      0.544    0.916     0.983     FALSE
                 0.534    0.044    0.687      0.534    0.601      0.544    0.916     0.651     TRUE
Weighted Avg.    0.890    0.401    0.882      0.890    0.884      0.544    0.916     0.931

=== Confusion Matrix ===

    a    b   <-- classified as
 9959  463 |    a = FALSE
  890 1018 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (4,1)



```
Number of performed Iterations: 10


Cost Matrix
 0 1
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10770               87.3479 %
Incorrectly Classified Instances      1560               12.6521 %
Kappa statistic                          0.5883
Total Cost                            1560
Average Cost                             0.1265
K&B Relative Info Score                 13.0626 %
K&B Information Score                 1001.1395 bits      0.0812 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             5686.0741 bits      0.4612 bits/instance
Complexity improvement     (Sf)       1978.0982 bits      0.1604 bits/instance
Mean absolute error                      0.1928
Root mean squared error                  0.3096
Relative absolute error                 73.6972 %
Root relative squared error             85.6098 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.886    0.195    0.961      0.886    0.922      0.602    0.916     0.983     FALSE
                 0.805    0.114    0.564      0.805    0.663      0.602    0.916     0.656     TRUE
Weighted Avg.    0.873    0.182    0.900      0.873    0.882      0.602    0.916     0.932

=== Confusion Matrix ===

    a    b   <-- classified as
 9234 1188 |    a = FALSE
  372 1536 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (4,2)



```
Number of performed Iterations: 10


Cost Matrix
 0 2
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10848               87.9805 %
Incorrectly Classified Instances      1482               12.0195 %
Kappa statistic                          0.5889
Total Cost                            1482
Average Cost                             0.1202
K&B Relative Info Score                 26.967  %
K&B Information Score                 2066.7998 bits      0.1676 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4880.0938 bits      0.3958 bits/instance
Complexity improvement     (Sf)       2784.0784 bits      0.2258 bits/instance
Mean absolute error                      0.1685
Root mean squared error                  0.289
Relative absolute error                 64.3857 %
Root relative squared error             79.9143 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.903    0.244    0.953      0.903    0.927      0.596    0.914     0.981     FALSE
                 0.756    0.097    0.587      0.756    0.661      0.596    0.914     0.660     TRUE
Weighted Avg.    0.880    0.222    0.896      0.880    0.886      0.596    0.914     0.931

=== Confusion Matrix ===

    a    b   <-- classified as
 9406 1016 |    a = FALSE
  466 1442 |    b = TRUE
```

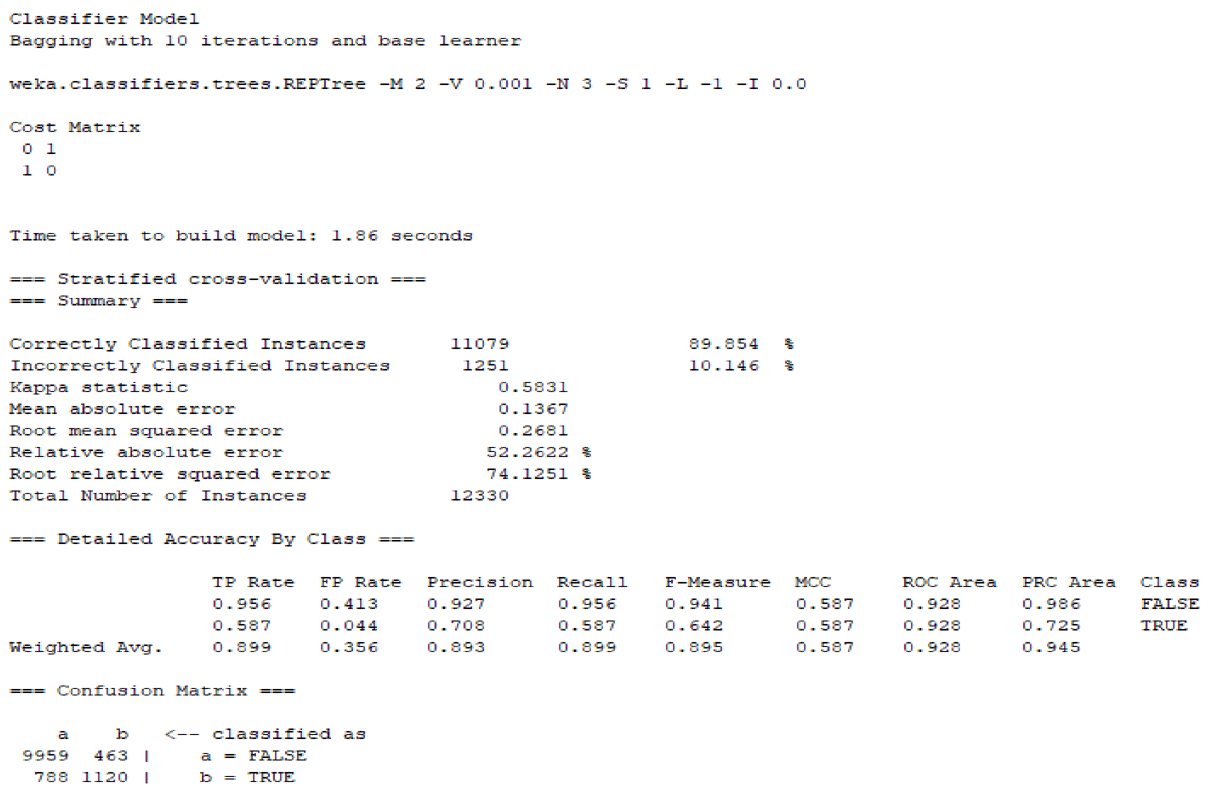
Boosting (AddBoostM1 Cost) (4,3)

```
Number of performed Iterations: 10


Cost Matrix
 0 3
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10943               88.751  %
Incorrectly Classified Instances      1387               11.249  %
Kappa statistic                          0.5896
Total Cost                            1387
Average Cost                             0.1125
K&B Relative Info Score                 33.7104 %
K&B Information Score                 2583.6201 bits      0.2095 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4622.8292 bits      0.3749 bits/instance
Complexity improvement     (Sf)       3041.343  bits      0.2467 bits/instance
Mean absolute error                      0.1532
Root mean squared error                  0.2827
Relative absolute error                 58.5583 %
Root relative squared error             78.1632 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.923    0.305    0.943      0.923    0.933      0.591    0.914     0.982     FALSE
                 0.695    0.077    0.622      0.695    0.657      0.591    0.914     0.647     TRUE
Weighted Avg.    0.888    0.270    0.893      0.888    0.890      0.591    0.914     0.930

=== Confusion Matrix ===

    a    b   <-- classified as
 9617  805 |    a = FALSE
  582 1326 |    b = TRUE
```

Boosting (AddBoostM1 Cost) (4,4)

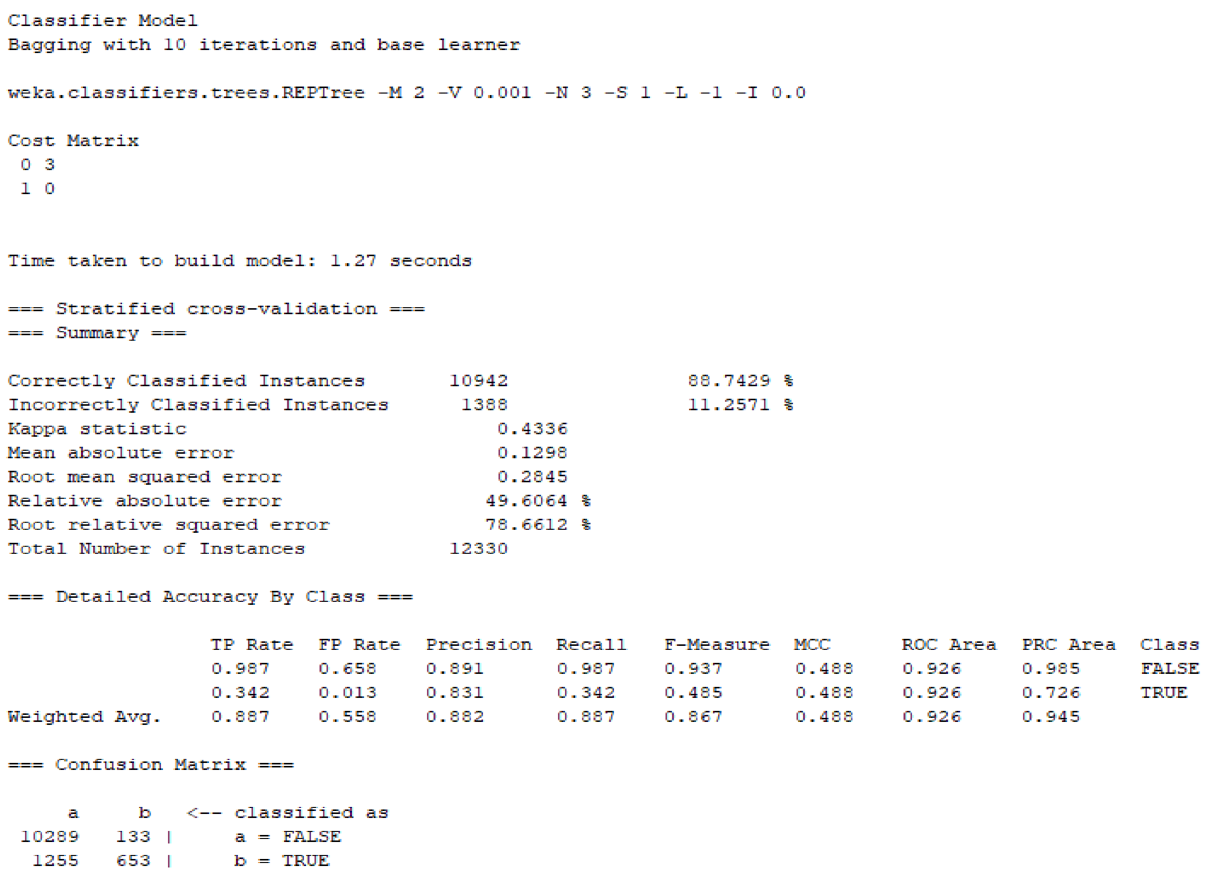
```
Number of performed Iterations: 10


Cost Matrix
 0 4
 4 0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10944               88.7591 %
Incorrectly Classified Instances      1386               11.2409 %
Kappa statistic                          0.5662
Total Cost                            1386
Average Cost                             0.1124
K&B Relative Info Score                 36.3197 %
K&B Information Score                 2783.6048 bits      0.2258 bits/instance
Class complexity | order 0            7664.1723 bits      0.6216 bits/instance
Class complexity | scheme             4538.355  bits      0.3681 bits/instance
Complexity improvement     (Sf)       3125.8173 bits      0.2535 bits/instance
Mean absolute error                      0.1467
Root mean squared error                  0.2794
Relative absolute error                 56.0737 %
Root relative squared error             77.2602 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.936    0.375    0.932      0.936    0.934      0.566    0.913     0.982     FALSE
                 0.625    0.064    0.640      0.625    0.633      0.566    0.913     0.649     TRUE
Weighted Avg.    0.888    0.327    0.887      0.888    0.887      0.566    0.913     0.930

=== Confusion Matrix ===

    a    b   <-- classified as
 9751  671 |    a = FALSE
  715 1193 |    b = TRUE
```

Bagging (1,1)



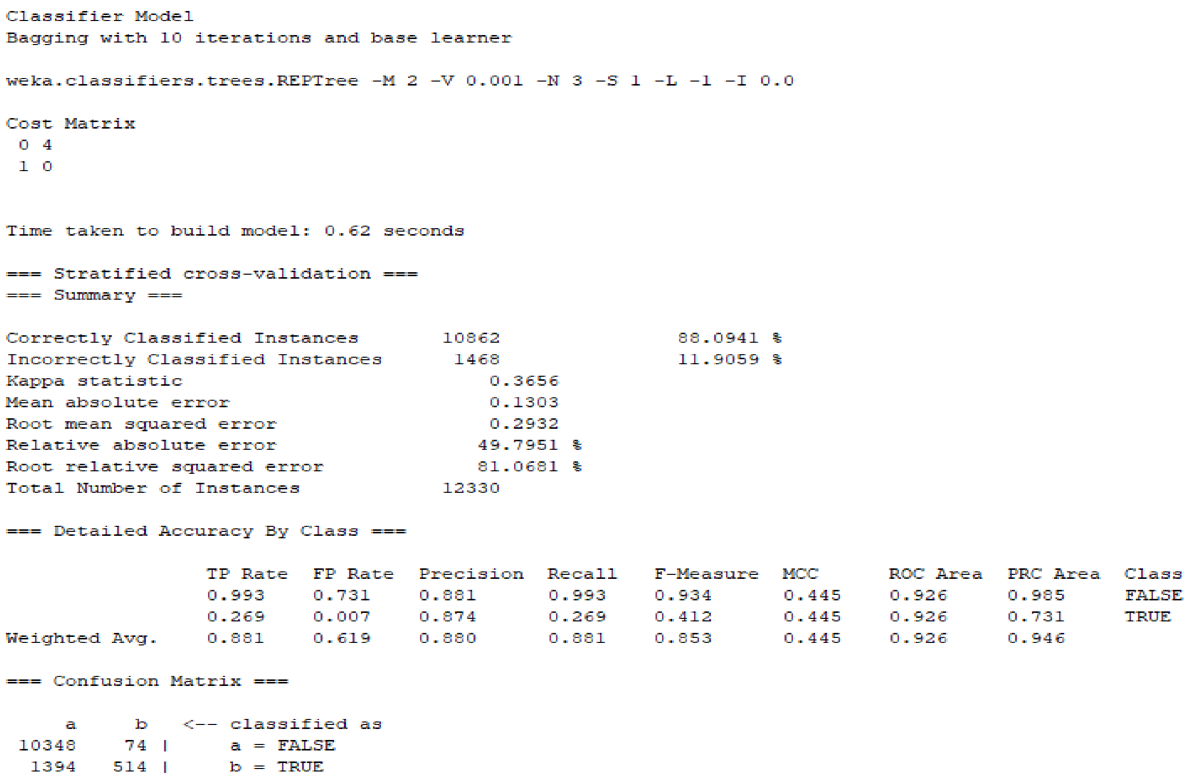
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 1
 1 0


Time taken to build model: 1.86 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11079               89.854  %
Incorrectly Classified Instances      1251               10.146  %
Kappa statistic                          0.5831
Mean absolute error                      0.1367
Root mean squared error                  0.2681
Relative absolute error                 52.2622 %
Root relative squared error             74.1251 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.956    0.413    0.927      0.956    0.941      0.587    0.928     0.986     FALSE
                 0.587    0.044    0.708      0.587    0.642      0.587    0.928     0.725     TRUE
Weighted Avg.    0.899    0.356    0.893      0.899    0.895      0.587    0.928     0.945

=== Confusion Matrix ===

    a    b   <-- classified as
 9959  463 |    a = FALSE
  788 1120 |    b = TRUE
```

Bagging (1,3)



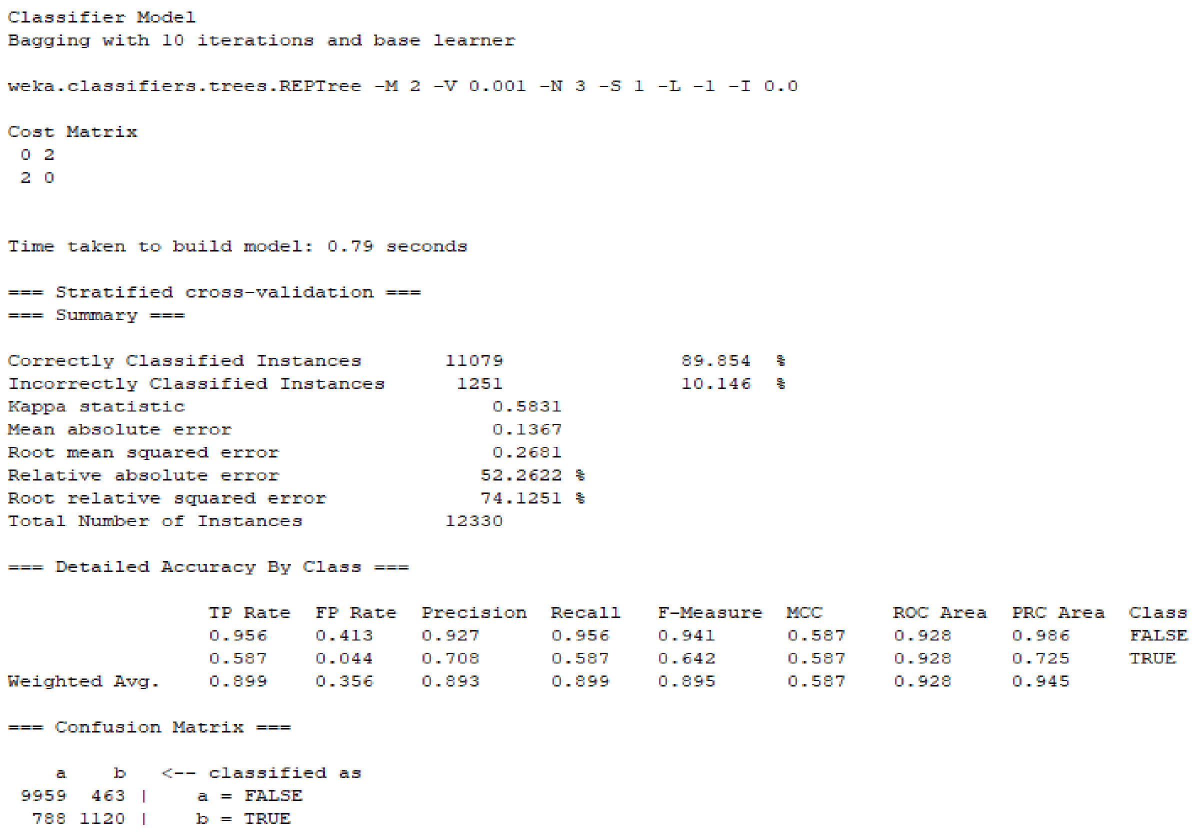
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 3
 1 0


Time taken to build model: 1.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10942               88.7429 %
Incorrectly Classified Instances      1388               11.2571 %
Kappa statistic                          0.4336
Mean absolute error                      0.1298
Root mean squared error                  0.2845
Relative absolute error                 49.6064 %
Root relative squared error             78.6612 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.987    0.658    0.891      0.987    0.937      0.488    0.926     0.985     FALSE
                 0.342    0.013    0.831      0.342    0.485      0.488    0.926     0.726     TRUE
Weighted Avg.    0.887    0.558    0.882      0.887    0.867      0.488    0.926     0.945

=== Confusion Matrix ===

     a     b   <-- classified as
 10289   133 |     a = FALSE
  1255   653 |     b = TRUE
```

Bagging (1,4)



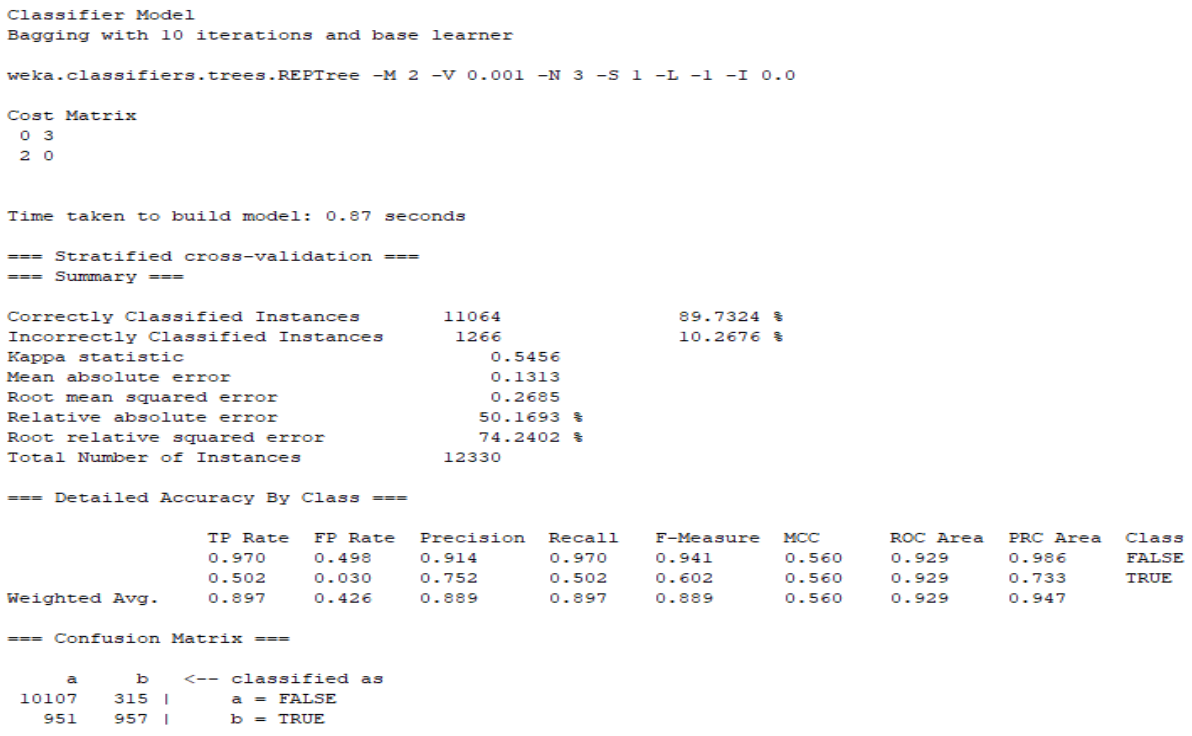
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 4
 1 0


Time taken to build model: 0.62 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10862               88.0941 %
Incorrectly Classified Instances      1468               11.9059 %
Kappa statistic                          0.3656
Mean absolute error                      0.1303
Root mean squared error                  0.2932
Relative absolute error                 49.7951 %
Root relative squared error             81.0681 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.993    0.731    0.881      0.993    0.934      0.445    0.926     0.985     FALSE
                 0.269    0.007    0.874      0.269    0.412      0.445    0.926     0.731     TRUE
Weighted Avg.    0.881    0.619    0.880      0.881    0.853      0.445    0.926     0.946

=== Confusion Matrix ===

     a     b   <-- classified as
 10348    74 |     a = FALSE
  1394   514 |     b = TRUE
```

Bagging (2,2)



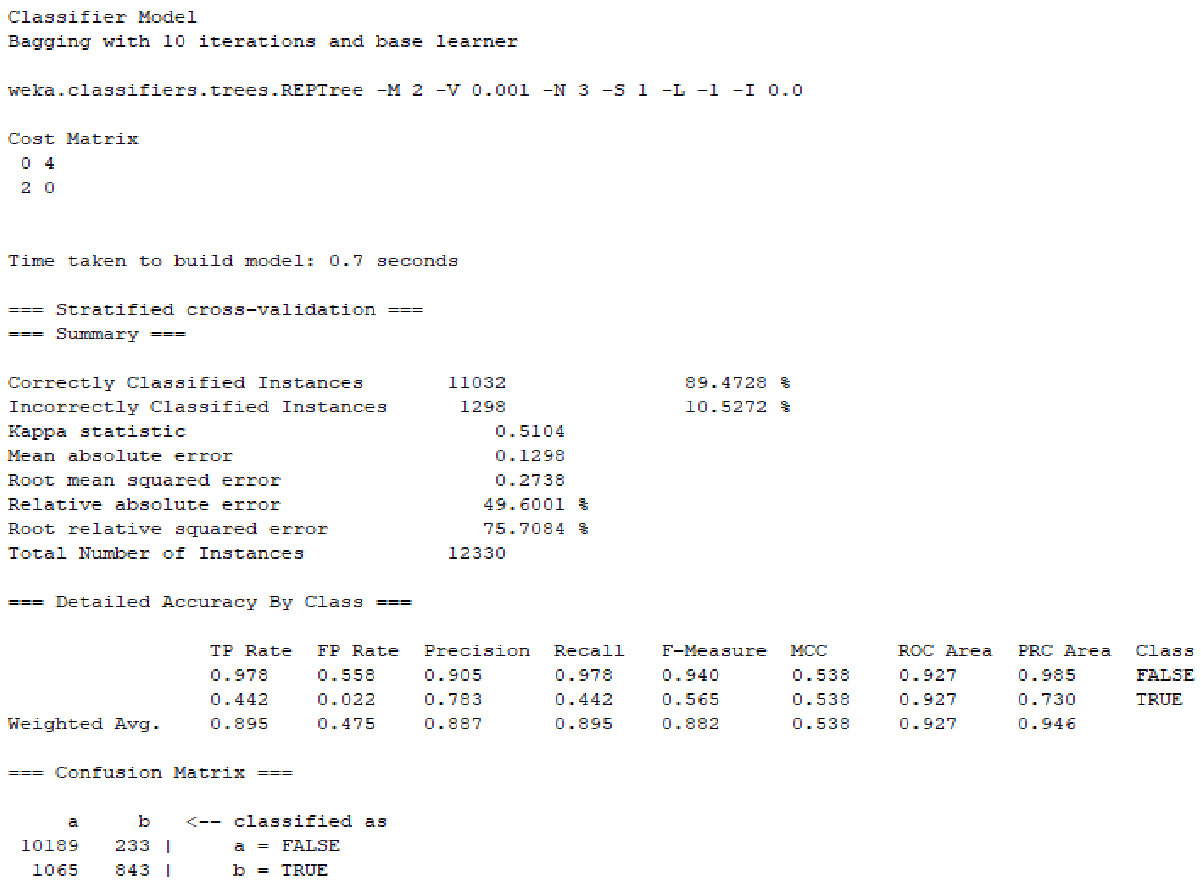
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 2
 2 0


Time taken to build model: 0.79 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11079               89.854  %
Incorrectly Classified Instances      1251               10.146  %
Kappa statistic                          0.5831
Mean absolute error                      0.1367
Root mean squared error                  0.2681
Relative absolute error                 52.2622 %
Root relative squared error             74.1251 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.956    0.413    0.927      0.956    0.941      0.587    0.928     0.986     FALSE
                 0.587    0.044    0.708      0.587    0.642      0.587    0.928     0.725     TRUE
Weighted Avg.    0.899    0.356    0.893      0.899    0.895      0.587    0.928     0.945

=== Confusion Matrix ===

    a    b   <-- classified as
 9959  463 |    a = FALSE
  788 1120 |    b = TRUE
```

Bagging (2,3)



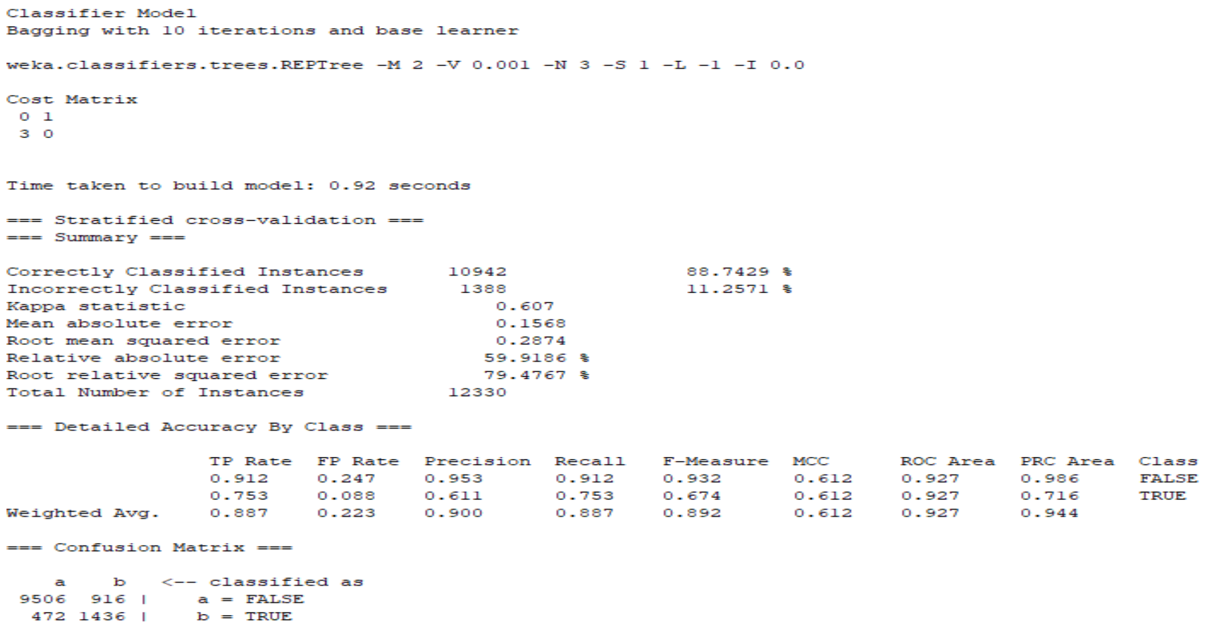
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 3
 2 0


Time taken to build model: 0.87 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11064               89.7324 %
Incorrectly Classified Instances      1266               10.2676 %
Kappa statistic                          0.5456
Mean absolute error                      0.1313
Root mean squared error                  0.2685
Relative absolute error                 50.1693 %
Root relative squared error             74.2402 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.970    0.498    0.914      0.970    0.941      0.560    0.929     0.986     FALSE
                 0.502    0.030    0.752      0.502    0.602      0.560    0.929     0.733     TRUE
Weighted Avg.    0.897    0.426    0.889      0.897    0.889      0.560    0.929     0.947

=== Confusion Matrix ===

     a     b   <-- classified as
 10107   315 |     a = FALSE
   951   957 |     b = TRUE
```

Bagging (2,4)



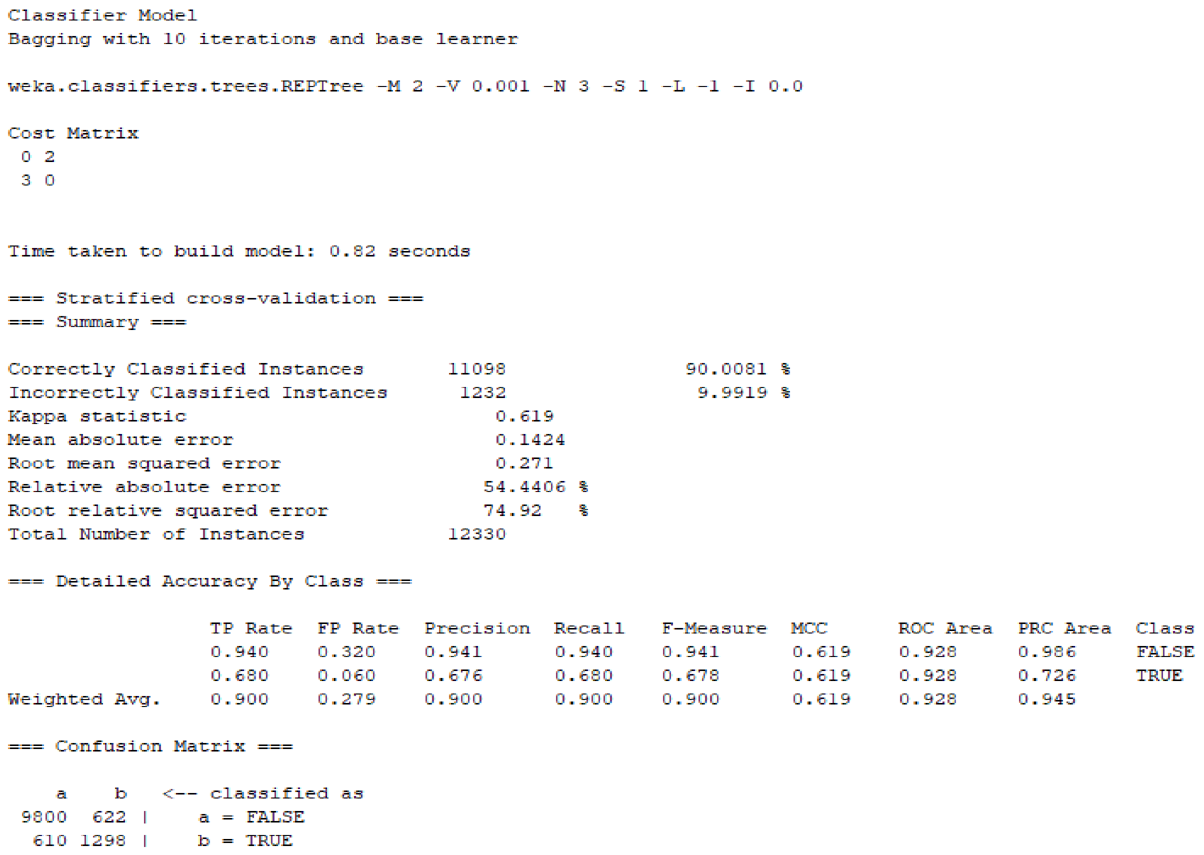
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 4
 2 0


Time taken to build model: 0.7 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11032               89.4728 %
Incorrectly Classified Instances      1298               10.5272 %
Kappa statistic                          0.5104
Mean absolute error                      0.1298
Root mean squared error                  0.2738
Relative absolute error                 49.6001 %
Root relative squared error             75.7084 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.978    0.558    0.905      0.978    0.940      0.538    0.927     0.985     FALSE
                 0.442    0.022    0.783      0.442    0.565      0.538    0.927     0.730     TRUE
Weighted Avg.    0.895    0.475    0.887      0.895    0.882      0.538    0.927     0.946

=== Confusion Matrix ===

     a     b   <-- classified as
 10189   233 |     a = FALSE
  1065   843 |     b = TRUE
```

Bagging (3,1)



```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 1
 3 0


Time taken to build model: 0.92 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10942               88.7429 %
Incorrectly Classified Instances      1388               11.2571 %
Kappa statistic                          0.607
Mean absolute error                      0.1568
Root mean squared error                  0.2874
Relative absolute error                 59.9186 %
Root relative squared error             79.4767 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.912    0.247    0.953      0.912    0.932      0.612    0.927     0.986     FALSE
                 0.753    0.088    0.611      0.753    0.674      0.612    0.927     0.716     TRUE
Weighted Avg.    0.887    0.223    0.900      0.887    0.892      0.612    0.927     0.944

=== Confusion Matrix ===

    a    b   <-- classified as
 9506  916 |    a = FALSE
  472 1436 |    b = TRUE
```

Bagging (3,2)



```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 2
 3 0


Time taken to build model: 0.82 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11098               90.0081 %
Incorrectly Classified Instances      1232                9.9919 %
Kappa statistic                          0.619
Mean absolute error                      0.1424
Root mean squared error                  0.271
Relative absolute error                 54.4406 %
Root relative squared error             74.92   %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.940    0.320    0.941      0.940    0.941      0.619    0.928     0.986     FALSE
                 0.680    0.060    0.676      0.680    0.678      0.619    0.928     0.726     TRUE
Weighted Avg.    0.900    0.279    0.900      0.900    0.900      0.619    0.928     0.945

=== Confusion Matrix ===

    a    b   <-- classified as
 9800  622 |    a = FALSE
  610 1298 |    b = TRUE
```

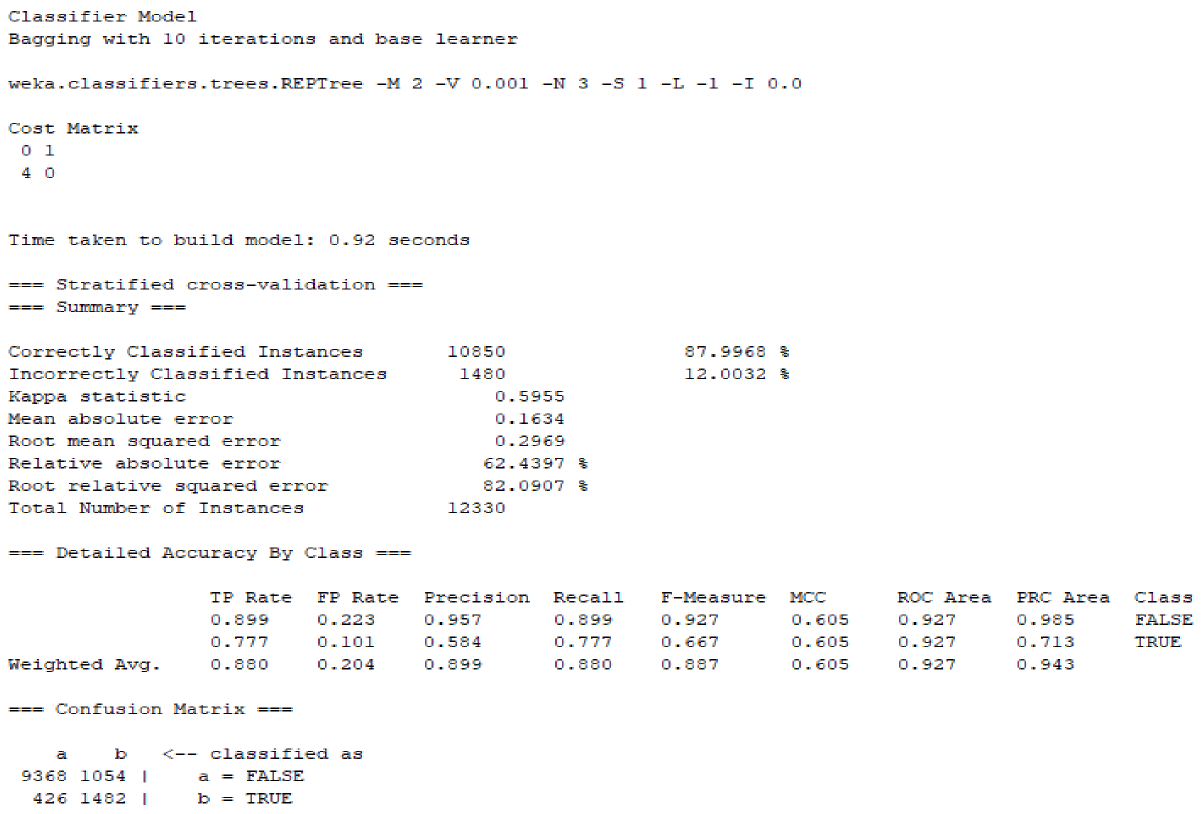
Bagging (3,3)

```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 3
 3 0


Time taken to build model: 0.94 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11079               89.854  %
Incorrectly Classified Instances      1251               10.146  %
Kappa statistic                          0.5831
Mean absolute error                      0.1367
Root mean squared error                  0.2681
Relative absolute error                 52.2622 %
Root relative squared error             74.1251 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.956    0.413    0.927      0.956    0.941      0.587    0.928     0.986     FALSE
                 0.587    0.044    0.708      0.587    0.642      0.587    0.928     0.725     TRUE
Weighted Avg.    0.899    0.356    0.893      0.899    0.895      0.587    0.928     0.945

=== Confusion Matrix ===

    a    b   <-- classified as
 9959  463 |    a = FALSE
  788 1120 |    b = TRUE
```

Bagging (3,4)

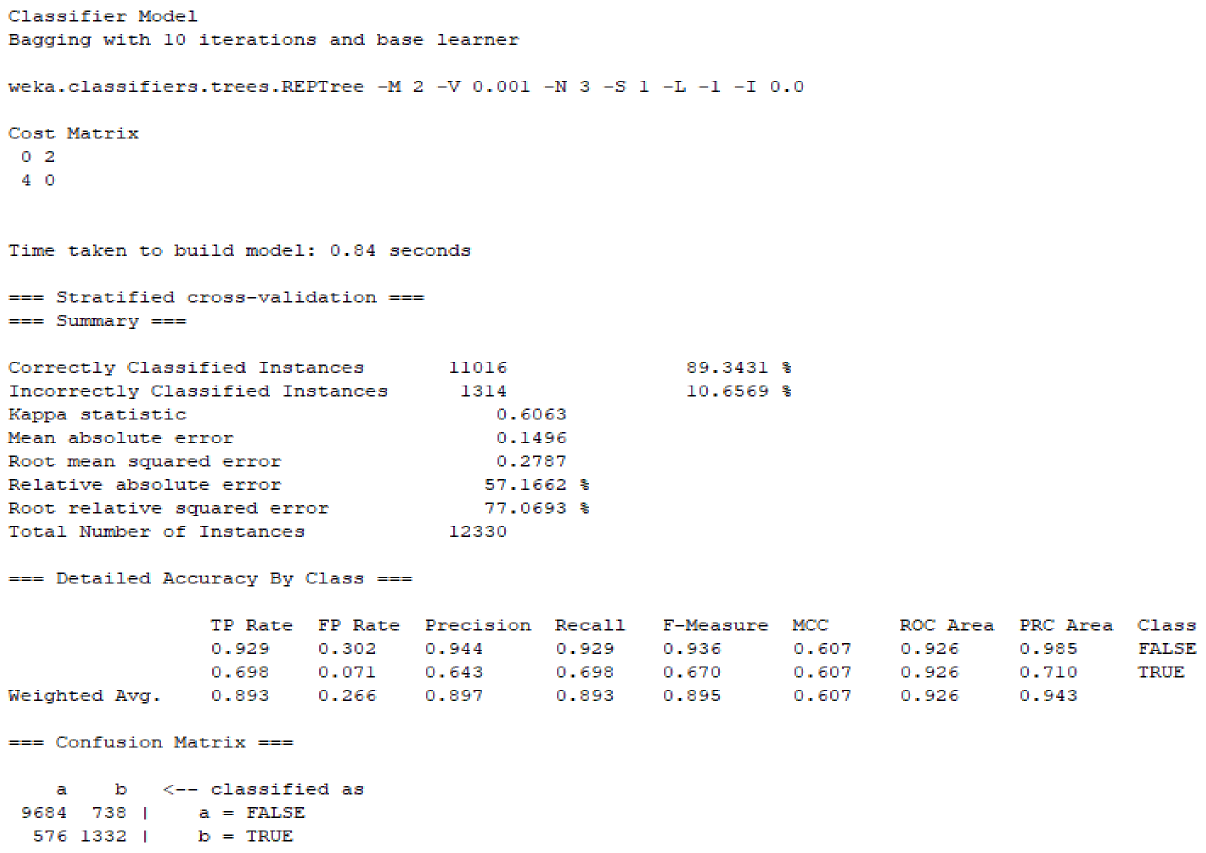
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 4
 3 0


Time taken to build model: 0.82 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11093               89.9676 %
Incorrectly Classified Instances      1237               10.0324 %
Kappa statistic                          0.5686
Mean absolute error                      0.1333
Root mean squared error                  0.2688
Relative absolute error                 50.9318 %
Root relative squared error             74.3201 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.966    0.460    0.920      0.966    0.942      0.578    0.927     0.985     FALSE
                 0.540    0.034    0.742      0.540    0.625      0.578    0.927     0.724     TRUE
Weighted Avg.    0.900    0.394    0.892      0.900    0.893      0.578    0.927     0.945

=== Confusion Matrix ===

     a     b   <-- classified as
 10063   359 |     a = FALSE
   878  1030 |     b = TRUE
```

Bagging (4,1)



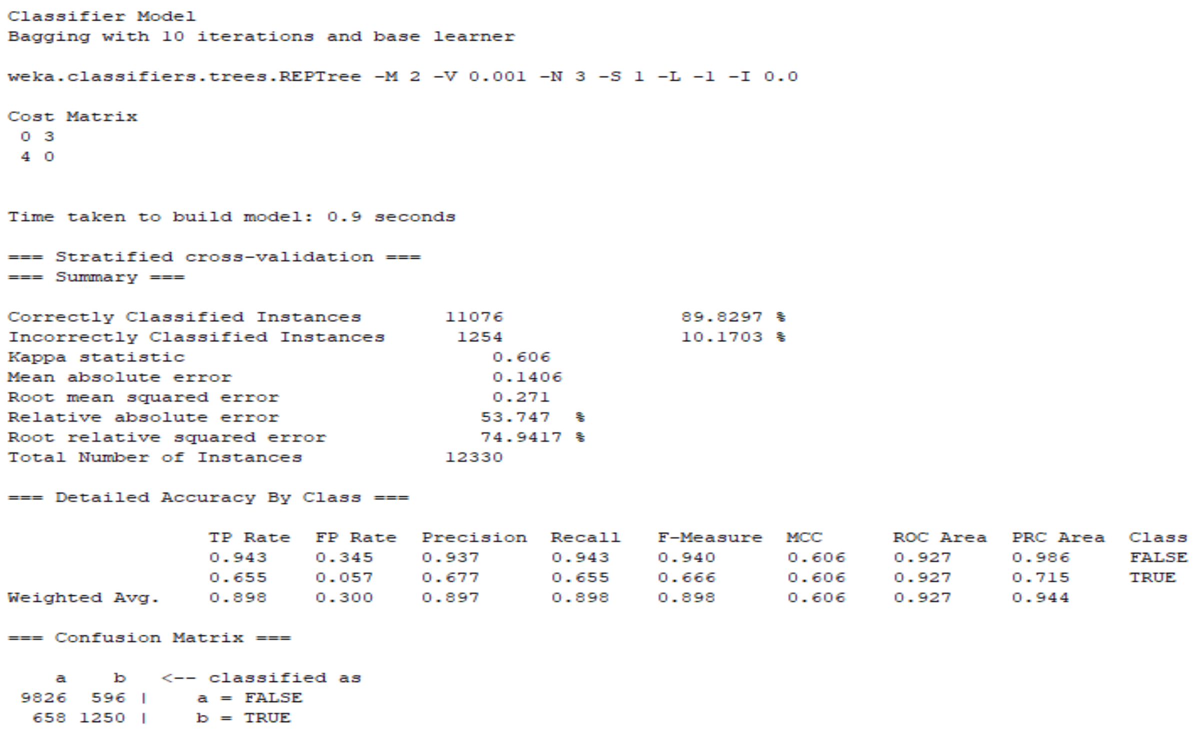
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 1
 4 0


Time taken to build model: 0.92 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       10850               87.9968 %
Incorrectly Classified Instances      1480               12.0032 %
Kappa statistic                          0.5955
Mean absolute error                      0.1634
Root mean squared error                  0.2969
Relative absolute error                 62.4397 %
Root relative squared error             82.0907 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.899    0.223    0.957      0.899    0.927      0.605    0.927     0.985     FALSE
                 0.777    0.101    0.584      0.777    0.667      0.605    0.927     0.713     TRUE
Weighted Avg.    0.880    0.204    0.899      0.880    0.887      0.605    0.927     0.943

=== Confusion Matrix ===

    a    b   <-- classified as
 9368 1054 |    a = FALSE
  426 1482 |    b = TRUE
```

Bagging (4,2)



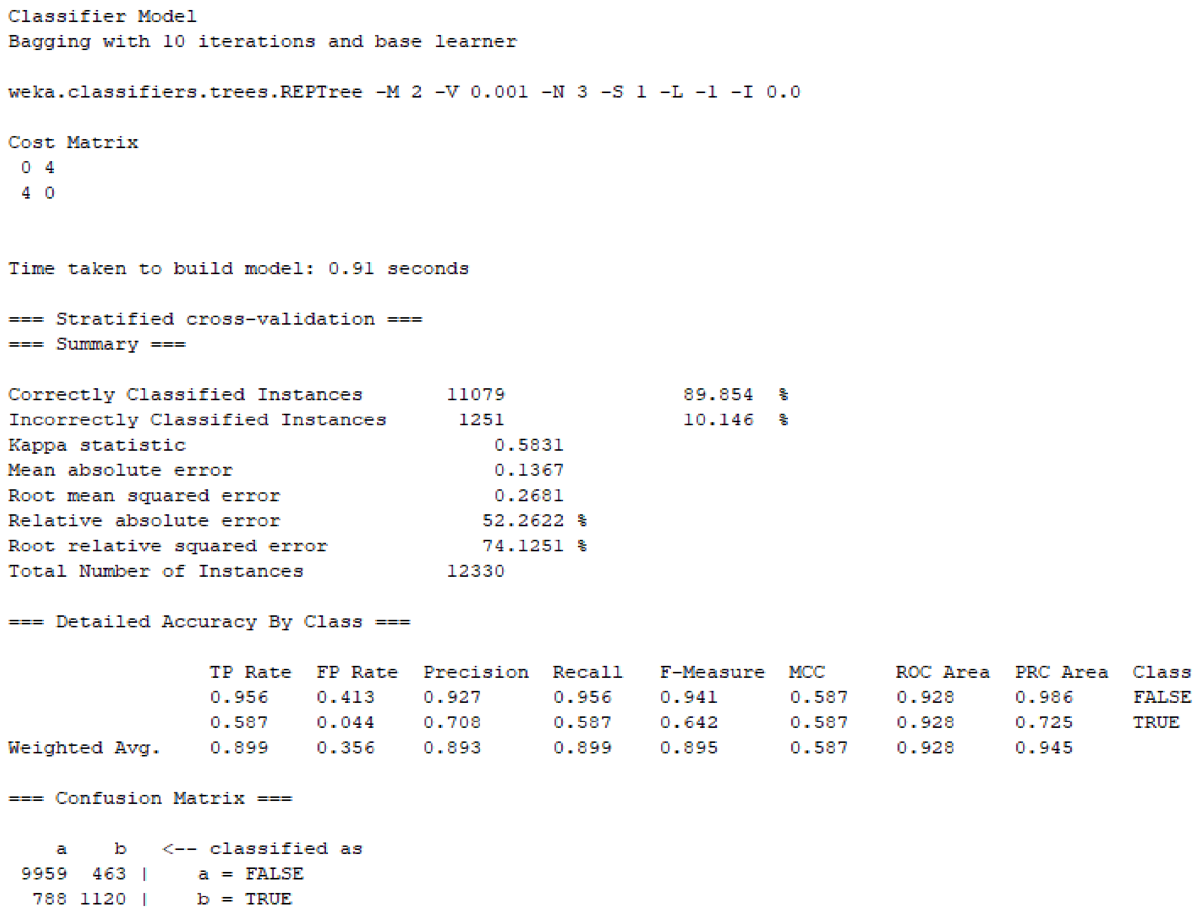
```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 2
 4 0


Time taken to build model: 0.84 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11016               89.3431 %
Incorrectly Classified Instances      1314               10.6569 %
Kappa statistic                          0.6063
Mean absolute error                      0.1496
Root mean squared error                  0.2787
Relative absolute error                 57.1662 %
Root relative squared error             77.0693 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.929    0.302    0.944      0.929    0.936      0.607    0.926     0.985     FALSE
                 0.698    0.071    0.643      0.698    0.670      0.607    0.926     0.710     TRUE
Weighted Avg.    0.893    0.266    0.897      0.893    0.895      0.607    0.926     0.943

=== Confusion Matrix ===

    a    b   <-- classified as
 9684  738 |    a = FALSE
  576 1332 |    b = TRUE
```

Bagging (4,3)



```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 3
 4 0


Time taken to build model: 0.9 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11076               89.8297 %
Incorrectly Classified Instances      1254               10.1703 %
Kappa statistic                          0.606
Mean absolute error                      0.1406
Root mean squared error                  0.271
Relative absolute error                 53.747  %
Root relative squared error             74.9417 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.943    0.345    0.937      0.943    0.940      0.606    0.927     0.986     FALSE
                 0.655    0.057    0.677      0.655    0.666      0.606    0.927     0.715     TRUE
Weighted Avg.    0.898    0.300    0.897      0.898    0.898      0.606    0.927     0.944

=== Confusion Matrix ===

    a    b   <-- classified as
 9826  596 |    a = FALSE
  658 1250 |    b = TRUE
```

Bagging (4,4)



```
Classifier Model
Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Cost Matrix
 0 4
 4 0


Time taken to build model: 0.91 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11079               89.854  %
Incorrectly Classified Instances      1251               10.146  %
Kappa statistic                          0.5831
Mean absolute error                      0.1367
Root mean squared error                  0.2681
Relative absolute error                 52.2622 %
Root relative squared error             74.1251 %
Total Number of Instances            12330

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.956    0.413    0.927      0.956    0.941      0.587    0.928     0.986     FALSE
                 0.587    0.044    0.708      0.587    0.642      0.587    0.928     0.725     TRUE
Weighted Avg.    0.899    0.356    0.893      0.899    0.895      0.587    0.928     0.945

=== Confusion Matrix ===

    a    b   <-- classified as
 9959  463 |    a = FALSE
  788 1120 |    b = TRUE
```