# ***CIS 6930: Introduction to Data Mining (Fall 2017)***

# Project Report – AVAZU CTR Prediction

By Jay Shah (1461-3930)
Vivek Goyal (8388-6402)
Avinash Anand (3592-6793)
Arnav Upadhyaya (1111-4048)
Mohit Mewara (8413-2114)

12/11/2017

## Abstract

Online Advertising allows advertisers to bid and pay for measurable user responses which makes them to earn lot of money through that. With over millions of users being active online on thousands of social networking sites, e-commerce websites, news websites predicting whether the adds appear on their display will be clicked or not is a very challenging task. This allows advertisers to deliver user specific advertisements to earn more. In this project, we will perform different classification techniques on Avazu dataset to predict whether the ads will be clicked or not.

## Introduction

Digital advertising is multibillion dollar industry and is growing dramatically each year. It is important to display user specific ads to increase the chances of ads being clicked. CTR stands for Click Through Rate. CTR is a very important measure of evaluating performance of an ad. As a global company in advertisement sector, it is essential for Avazu to know the performance of the various ads placed by the company. To achieve this goal, Avazu created a competition on Kaggle, to build a prediction model which best fits the data collected by the company.

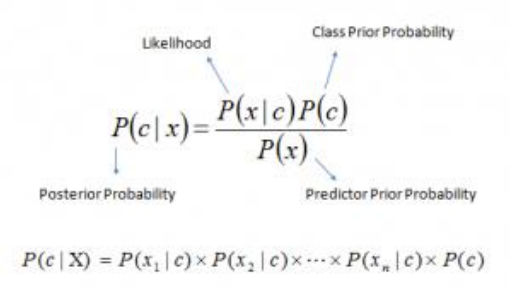
The company provided a total of 11 days of Avazu data, which contains various specifications of the ad, details of device where it is posted, and the type of connection used by the device. Two separate files, train.csv (5.87 GB) which contained the details to predict the model, and test.csv (673 MB) which contained data to test the predicted model in the previous step. In the modified version of the project that we decided to work on, we decided to implement Classification prediction models, so we are not using the test.csv in out project. Rather we divided the train.csv in two parts, 70% of the data is treated as new training data, and the rest 30% data is used for testing the accuracy of the generated model.

Different classification techniques help us to predict whether the ads will be clicked or not. It uses input set of numerical features such as device id, device ip, site id, application domain, application id, etc. and output the classified data. When the new data is given as an input it will classify the data accordingly in 0 or 1 where 0 means that ads will not be clicked and 1 means ads will be clicked.

There are 4 types of classification techniques that we have used in our project.

1. **Naïve Bayes Classification**: -
   Naive Bayes classifier is a classifier based on Bayes Theorem. It follows the principle that every feature being classified is independent of the values of the other features regardless of any correlations between them. Features however are not always independent which is often seen as a shortcoming of naive Bayes algorithm. It is a probabilistic model, which means that it calculates the probability of each category for a sample, and then outputs the category with the highest one.

Naive Bayes Classifier is easy to build and highly scalable. Along with simplicity, it can sometimes outperform highly sophisticated model. Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:




a. P(c|x) is the posterior probability of class (c, target) given features (x, attributes).
b. P(c) is the prior probability of class.
c. P(x|c) is the likelihood which is the probability of features given class.
d. P(x) is the prior probability of features.

2. **Support Vector Machine**: -

Support Vector Machine is a supervised Machine Learning Algorithm which is used for different classification challenges. Support Vector Machine is a binary classifier which classifies the data in 2 classes at a time. It can be visualized as a n-dimensional space where each data point represents each record and n is total number of numerical features. After plotting the data, one hyper plane will be used to separate the data. When the new data is given as an input, algorithm will plot the data in n-dimensional space and hyperplane will decide in which class the new data is classified.

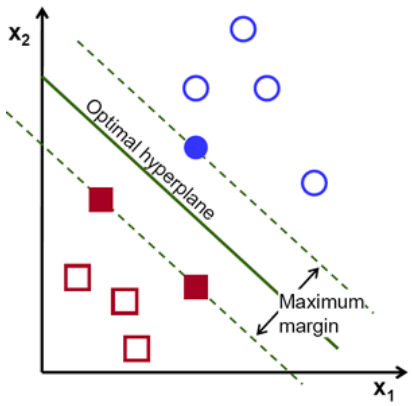
If hyperplane is dividing the data, then following equation must be satisfied.

$Y_i * ((W*X_{i)} + b) \geq 0$ for i = 1,2, 3..., n

- $Y_i$ is the number of classes
- W is a vector which is normal to hyperplane
- $X_i$ data point

Here W and b are unknown parameters of the model and different combination of those values can decide the separating hyperplane.

Choosing hyperplane is a challenging task. See below figure.



Hyperplane is chosen such that it maximizes the margin between red squared data and blue circled data which requires to minimize the value of W. Support Vector Machine converges to LaGrange's multiplier formula of quadratic equation where different combinations of W and b are given as an input to find an optimal hyperplane.

3. **C 4.5 Classification**: -
   C4.5 is a decision tree based classifier. Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label.

   Decision tree based classification is popular because the construction of decision tree classifiers do not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction is simple and fast. In general, decision tree classifiers have good accuracy. However, successful use may depend on the data at hand. Decision tree induction algorithms have been used for classification in many application areas, such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. Decision trees are the basis of several commercial rule induction systems.

4. **Random Forest Classification**: -
   Random Forest classification is an extension of bagging. It fits n number of decision trees on various samples of the data and then compute the average among all decision trees to improve the accuracy. Apart from building different decision trees using sub-samples, it also constrains the feature that can be used to build the trees, forcing trees to be different.

   Decision trees involve the greedy algorithm which selects the best split from the dataset at each iteration. This approach is susceptible to high variance if we did not prune the tree. The variance can be reduced by generating multiple decision trees using different samples of the dataset. This approach is known as bagging. However, a limitation of bagging is that it uses the same algorithm to generate several trees, meaning that the splits would be similar in every tree. We can force the decision trees to be apart from each other by randomly picking different features at each split while creating the tree. This algorithm which is an extension of bagging is known as random forest algorithm.

# Background and Literature Review

Daniel Lowd proposes naive Bayes models as an alternative to Bayesian networks for general probability estimation tasks. He showed that others take similar time to learn and are similarly accurate, but naive Bayes inference is orders of magnitude faster [3]. Daniel D. Lewis explained that recent comparisons of learning methods for text recognition have been somewhat less favorable to Naïve Bayes models while still showing respectable effectiveness and this may be due to the large amount of training data available in categorization data favors algorithm which produce more complex classifiers [4].

V N Vapnik described that the Support Vector Machine is based on Structural Risk Minimization(SRM) principle rooted in the statistical learning theory. Generalization abilities are increased in this algorithm and SRM is achieved through a minimization of generalization error [1]. Traditional Classifiers are trained in a way that empirical risks are minimized but generalization errors remain in their algorithms which is not the case with Support Vector Machines. The Performance of SVM is reviewed in various classification tasks in Christiani and Shawe-Taylor [2].

Sandro Morasca has proposed usage of continuous attributes in C4.5 classification algorithm. Classification trees have been successfully used in several application fields. However, continuous attributes cannot be used directly when building classification trees, but they must be first discretized with clustering techniques, which require some

degree of subjectivity. They propose an approach to build classification trees that does not require the discretization of the continuous attributes. The approach is an extension of existing methods for building classification trees and is based on the information gain yielded by discrete and continuous attributes [5].

Son T. Vuong proposed that the random forest classification can be used for detecting malware in android devices. They focus on detection accuracy as the free parameters of the Random Forest algorithm such as the number of trees, depth of each tree and number of random features selected are varied. Their experiment shows that Random Forest performs very well with an accuracy of over 99 percent in general [6]. Lei Nie proposes an algorithm for analyzing very high dimensional data with multiple classes whose well-known representative data is image data. Experimental results have demonstrated that the proposed method could generate a random forest model with higher performance than the random forest [7].

# Data source and Dataset Description

As already mentioned, this problem and the corresponding dataset is taken from Kaggle, where it is a competition managed by Avazu. The link to download the dataset is - https://www.kaggle.com/c/avazu-ctr-prediction.

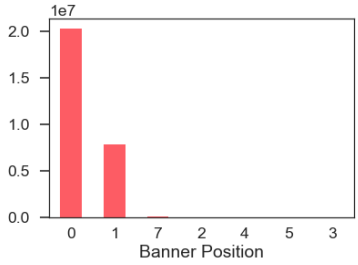
Below is the description of dataset given in the training file (the number in the brackets are the number of unique value for respective attributes) –

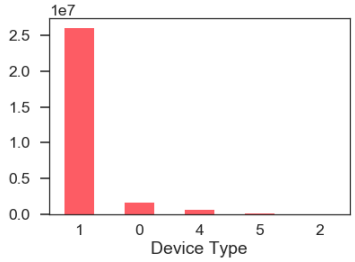
id – Ad id (28300275)
click – 0/1, 1 signifying click, 0 signifying non-click
hour – YYMMDDHH format, so 14091123 means 23:00 on Sept. 11, 2014 UTC
banner_pos – position of banner in application (7)
site_id – ID given to site where ad is posted (4555)
site_domain – Domain of the site where ad is posted (7087)
site_category – Each site is divided in various categories (26)
app_id – ID of the application where ad is posted (7926)
app_domain – Domain of the application where ad is posted (510)
app_category – Application is divided in various categories (34)
device_id – Each device has a unique ID (2134670)
device_ip – IP of the device used by user (5604672)
device_model – Model of the device used by user (7944)
device_type – Each device is divided in various device types (5)
device_conn_type – Connection type of the device (4)
C1, C14-C21 – Anonymized variables given by the company

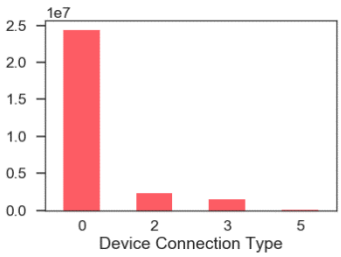
# Data Visualization and Preprocessing

## Visualization -

To continue with the visualization, we need data which are less in unique numbers, and as per the above data, there are only few attributes (namely banner_pos, device_type and device_conn_type) which can be used to plot a graph. So, we plotted the graph for the 3 attributes. The graphs are –



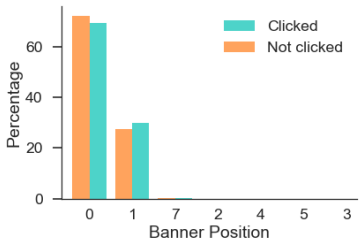



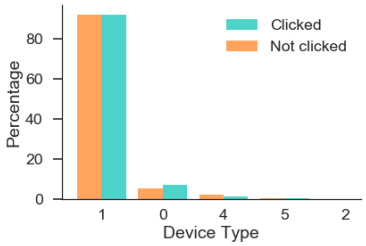

Following observations can be made from the above graphs –

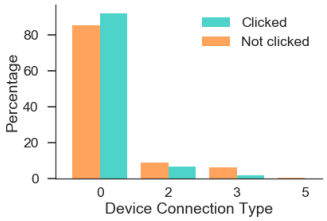
1.) Most of the data is for the banner position 0 and 1.

2.) Device type 1 fetches most of the data.
3.) Focus is on a specific connection type, which is 0.

Now that we have these graphs, next we tried to find the distribution of clicks and non-clicks in the above graph, to check if the focus on the types are correct or not. So, we generated the below graphs –




The one and most important observation that can be made from above 3 graphs is that doesn't matter the banner position, device type or the device connection type, ratio of clicks to non-clicks for the three features is almost the same. This means that we cannot ignore the low values as they will also contribute in the model generation

## Preprocessing –

After the Visualization step, the next step comes Preprocessing. The very first data change we did was dividing the training data in new training (70% of original training data) and test (30% of original training data). The reason for this is already explained earlier.
Next comes changing the data to fit the model. The very first part is Advertisement id, which counts to approx. 28 mil data out of approx. 40 mil data. We can infer that id is not going to help us in creating the model, as there are too many unique values and usually there is no relation between any of the id's. So, we dropped this column from the data.
One very important data that can be used in modelling is the time. There are times when the users are more active and probability to click an ad is more compared to when not many users are active. So, time could play a key role. Similarly, date could be a factor and could help in creating model. To achieve this, we removed the 'hour' attribute and extracted 'HH' and 'DD' to create two new attributes, which would be more helpful in creating models, than the pre-existing 'hour' attribute.

# Algorithm Description and Implementation

## 1. Naïve Bayes' Classification

Algorithm is described and implemented as follows.

1. Splitting the data into train-set and test-set.
2. Summarize data - summarize the properties in training dataset so that we can calculate probabilities and make predictions:
    1. Separate the training data-set instances by class value to calculate the statistics of each class.
        - Done by creating a map of each class value to list of instances that belong to the class
    2. Calculate mean and standard deviation for each attribute and class value combination.
        - First separate training dataset into instances grouped by class

- Use the zip function to group the values for each attribute across our data instances into their own lists

3. Predictions

    a. Calculate the probability that a given data instance belongs to each class, then select the class with the largest probability

    b. Given the known mean and standard deviation, Gaussian function is used to estimate the probability of a given attribute value.

    c. Combine the probabilities of all of the attribute values for a data instance by multiplying together the attribute probabilities for each class.

    d. Look for the largest probability and return the associated class

    e. Prediction of each data instance is compared to the class value in test dataset to obtain Accuracy.

## 2. Support Vector Machine

Support Vector Machine requires lots of computation to choose a combination of W and b which will give the optimal hyperplane. That is why it takes a lot of time if implemented on large dataset. We have implemented this algorithm on small chunk of the dataset to check its efficiency.

Algorithm is described and implemented as follows.

1. Splitting the dataset into size 100k and 20k respectively.
2. Identify the max feature value in the dataset and it is given as a vector W.
3. Visually we are trying to plot the data in n-dimensional feature space and then we separate the data by defining hyperplane between them. Equation of hyperplane is given by
4. $WX + b = 0$.
5. Where X is a feature vector
6. We give each feature vector as an input and try different combination of W and b which satisfies the above equation. This is the step where Support Vector Machine algorithm takes large amount of time.
7. We are decreasing W's value as we need to find minimum W to get optimal hyperplane.
8. We use these W and b values to classify test data.
9. Based on the classification we get the prediction whether ad will be clicked or not.
10. Calculate the accuracy based on the classification

## 3. C 4.5 Classification

Algorithm is described and implemented as follows.

1. Splitting the data into train-set(~16M) and test-set(~4M).
2. Preprocessing the dataset
    a. Convert attributes that are numeric to float
3. Constructing decision tree
    a. Let T be the set of training instances
    b. Choose an attribute that best differentiates the instances contained in T (Using Entropy and Information Gain Ratio)
        - Calculate the entropy for all training examples
        - positive and negative cases
            a. $p+$ = #pos/Tot  $p-$ = #neg/Tot
            b. $H(S) = -p+log2(p+) - p-log2(p-)$
        - Determine which *single* attribute best classifies the training examples using information gain.
        - For each attribute find:

$$Gain(S, A_i) = H(S) \quad - \sum_{v \in Values(A_i)} P(A_i = v) H(S_v)$$

        - Use attribute with greatest information gain as a root
    c. Create a tree node whose value is the chosen attribute
    d. Create child links from this node where each link represents a unique value for the chosen attribute
    e. Use the child link values to further subdivide the instances into subclasses
    f. Note: It iterates until, a tree is built from the entire training set
4. Classify test-set
5. Calculate Accuracy
6. Calculate true negative, true positive, false positive, false negative, accuracy, precision, recall, f-measure, and sensitivity.

## 4. Random Forest Classification

Algorithm is described and implemented as follows.

1. Splitting the data into train-set and test-set.
2. Randomly select “k” features from total features. We have considered “k” to be square root of total features
3. Using the “k” features and best split point, calculate the node
4. Recursively split the node into child nodes using the best split
5. Repeat 2 to 4 steps until “l” number of nodes has been reached
6. Repeat steps from 2 to 5 to build "n" trees

Prediction:

1. Use each decision tree to predict the outcome and stores the predicted outcome
2. Calculate the true negative, true positive, false positive, false negative, accuracy, precision, recall, f-measure, and sensitivity for each predicted target

# Results and Analysis

Here, Results are analyzed for each algorithm implemented and compared visually as below.

1. **Naïve Bayes’ classification**: -

It took approximately 15 mins for analyzing dataset.

| | **Condition Positive** | **Condition Negative** |
|---|---|---|
| Predicted Condition Positive | 22.05% | 37.1% |
| Predicted Condition Negative | 4.9% | 35.95% |

Accuracy = 58%
Precision = TP/(TP+FP) = 0.37
Recall = TP/(TP+FN) = 0.82
F-measure = 2*Precision*Recall/(Precision + Recall) = 0.51

2. **Support Vector Machine**: -

It took approximately 180 mins for analyzing the dataset.

| | **Condition Positive** | **Condition Negative** |
|---|---|---|
| Predicted Condition Positive | 33.33% | 41.66% |
| Predicted Condition Negative | 8.33% | 16.66% |

Accuracy = 49%
Precision = TP/(TP+FP) = 0.44
Recall = TP/(TP+FN) = 0.66
F-measure = 2*Precision*Recall/(Precision + Recall) = 0.528

3. **C 4.5 Classification**: -

It took approximately 18 hours for analyzing the dataset.

| | **Condition Positive** | **Condition Negative** |
|---|---|---|
| Predicted Condition Positive | 52.40% | 7.35% |
| Predicted Condition Negative | 9.47% | 29.78% |

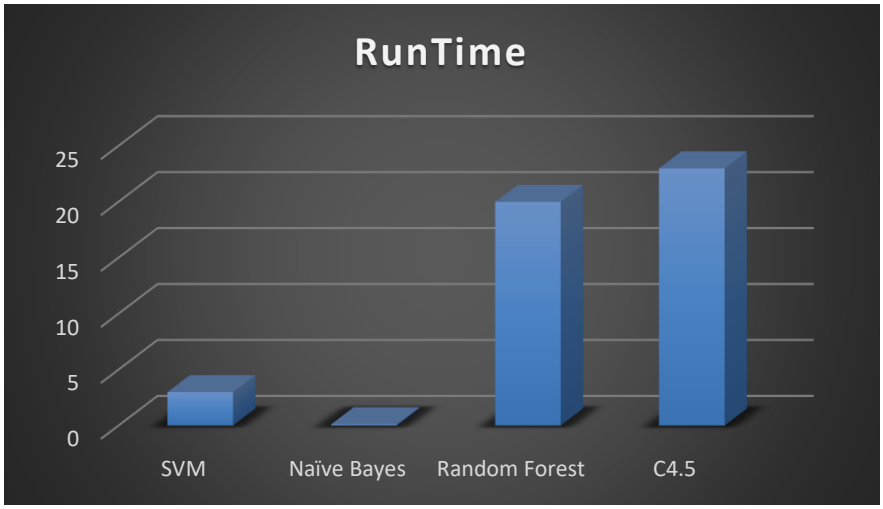
Accuracy = 82.18%
Precision = TP/(TP+FP) = 0.84
Recall = TP/(TP+FN) = 0.87
F-measure = 2*Precision*Recall/ (Precision + Recall) = 0.85

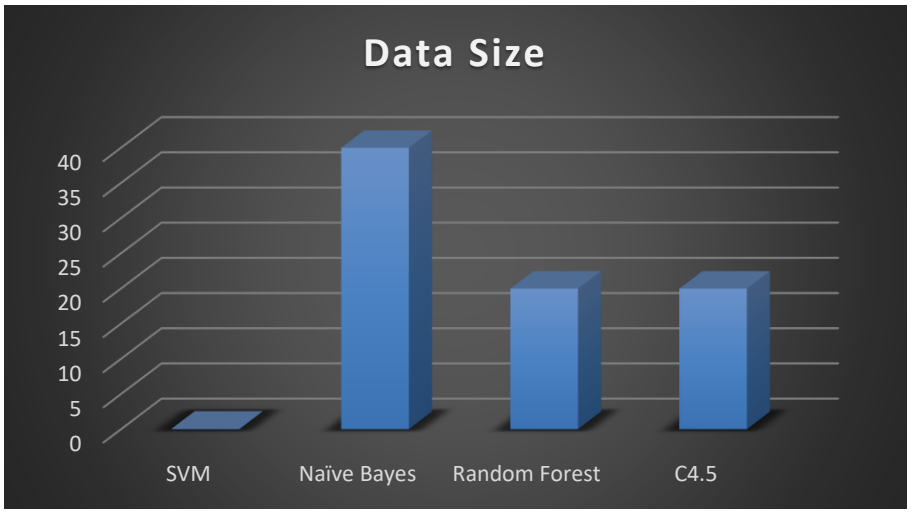
4. **Random Forest Classification**: -

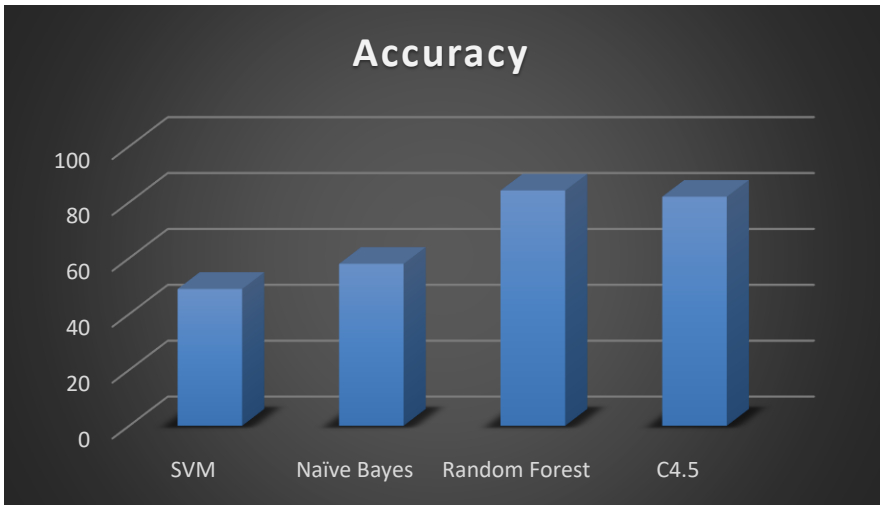
It took approximately 18 hours for analyzing the dataset.

| | Condition Positive | Condition Negative |
|---|---|---|
| Predicted Condition Positive | 52.22% | 5.35% |
| Predicted Condition Negative | 10.42% | 22.0% |

Accuracy = 84.22%
Precision = TP/(TP+FP) = 0.37
Recall = TP/(TP+FN) = 0.82
F-measure = 2*Precision*Recall/ (Precision + Recall) = 0.51






## Limitations and Future Work

There are certain limitations of the above algorithms in the terms of data processing and calculations. There were certain attributes namely C1 and C14 to C21 which were given anonymous. Through some research and forums, we figured out that two of them could be banner length and breadth, but we are not sure about all of them. As a reason of this, these attributes were considered in algorithmic calculations needless of their importance. Also, some of the attributes were filtered out on individual algorithm level because of non-numeric format. These considerations could have possibly reduced accuracies and performances of our algorithms to a certain level but even then, we kept our algorithmic performance on a positive side. In the probability based algorithm, we came across floating point underflow in python and assigned a hardcoded value if that happened. As a future work, these dependencies could be removed by using a numeric map for non-numeric entries and taking logarithmic calculation wherever floating-point underflow happens. Also, somewhat more research could provide a better view of what these anonymous columns were, and a better judgement could be made on attribute selection.

## Conclusion

The decision tree algorithm – C 4.5 and Random Forest gives the highest accuracy for the given dataset. However, the time required to train the decision tree is a lot and as the features or data size increases, the time increases proportionally. Although Naïve Bayes accuracy is not the highest, but its runtime is very less compared to all other algorithms. Considering both run time and accuracy as the performance measure, Naïve Bayes performance is highest for the given dataset. Also, the Naïve Bayes scale well with the increase in data size.

# References


[1] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York (1995)

[2] N. Cristianini, N.J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, 2000

[3] Lowd, Daniel, and Pedro Domingos. "Naive Bayes models for probability estimation." Proceedings of the 22nd international conference on Machine learning. ACM, 2005.

[4] Lewis, David D. "Naive (Bayes) at forty: The independence assumption in information retrieval." European conference on machine learning. Springer, Berlin, Heidelberg, 199

[5] Sandro Morasca. " A proposal for using continuous attributes in classification trees." Proceedings of the 14th international conference on Software engineering and knowledge engineering. SEKE '02

[6] Mohammed S. Alam, Son T. Vuong. “Random Forest Classification for Detecting Android Malware". IEEE and Internet of Things (iThings/CPSCom), 2013

[7] Baoxun Xu, Yunming Ye, Lei Nie. “An improved random forest classifier for image classification”. IEEE International Conference on Information and Automation, 2012