International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9, Issue-7, May 2020

Using Pyspark Environment for Solving a Big Data Problem: Searching for Supersymmetric Particles

Mourad Azhari, Abdallah Abarda, Badia Ettaki, Jamal Zerouaoui, Mohamed Dakkon

Abstract: ***Supersymmetry theory predicts that every particle in the standard model has a superpartner particle with a different mass. The Classification Problem of Supersymmetric Particles in High-Energy represents a major challenge for physicists. This paper aims to resolve the Big data Classification Problem in the area of Supersymmetric Particles using the Apache Spark Environment with the "MLlib" library. This contribution attempts to explore the performance of Machine Learning methods in the context of large data such as a "Susy" dataset, collected from the UCI Machine Learning repository. In this work, the performance is measured using three metrics: Accuracy, Area Under Curve (AUC), and training Computation Time (CT). The results are promising and show that the Gradient Boosted Tree (GBT) classifier achieves a high accuracy score (79%). While the Logistic Regression (LR) algorithm realizes a well AUC score (86%).***

Keywords: ***Machine Learning methods, performance, Spark Environment, Pyspark, Supersymmetric Particles.***

I. INTRODUCTION

Supersymmetric particle search is a principal concern for the High-Energy Physics area. Machine learning methods are a powerful tool that can improve the performance of Supersymmetric Particle Classification. Several studies related to Exotic Particle discovery have been proposed in the particle physics domain. In this context, ATLAS studied the predictions of the Standard Model. With this in mind, Higgs Boson Machine Learning Challenge has been tenuous in 2014.

Revised Manuscript Received on May 30, 2020.
***** Correspondence Author
Mourad Azhari*, Laboratory of Engineering Sciences and Modeling, Faculty of Sciences, Ibn Tofail University, Campus BP 133, Kenitra, Morocco, Email: azharimourad@yahoo.fr.
Abdallah Abarda, Laboratoire de Modélisation Mathématiques et de Calculs Economiques, FSJES, Université Hassan 1er, Settat, Morocco, Email: abardabdallah@gmail.com.
Badia Ettaki, Laboratory of Research in Computer Science, Data Sciences and Knowledge Engineering, Department of Data, Content and knowledge Engineering School of Information Sciences Rabat, Morocco, Email: ettakibadia@yahoo.fr
Jamal Zerouaoui, Laboratory of Engineering Sciences and Modeling, Faculty of Sciences, Ibn Tofail University, Campus BP 133, Kenitra, Morocco, Email: j_zerouaoui@yahoo.fr.
Mohamed Dakkon, Département de Statistique et Informatique de Gestion, Université Abdelmalek Essaadi, Tétouan, Morocco, Email : m.dakkoun@gmail.com

Many algorithms were tested, among these classifiers, we cite the basic algorithms, advanced methods, and ensemble methods. The accuracy score obtained with the Gradient Boosting algorithm achieved 84\% and Support vector machines (SVM) with linear kernels attained 76\% [1,2]. Baldi et al. have employed the Deep Networks method on Higgs and "Susy" datasets [3,4,5]. In multivariate statistical analysis, Alves proposed Stacking Machine Learning algorithms to classify Higgs Boson, this approach exceeds the Boosted Decision Trees and Deep Neural Network applicable to particle physics [6]. This work proposes to resolve the Supersymmetric Classification Problem with four Machine Learning methods with Pyspark Environment:

- Logistic Regression;
- Decision Tree;
- Random Forest;
- Gradient Boosted Tree.

Hence, we compare the accuracy and AUC metrics of those Machine Learning methods using five million (5000000) instances of the "Susy" dataset.

This paper includes the following sections: The first will attempt to propose a brief relevant work to Exotic Particle detection and the second presents the Spark Environment with proposed Machine Learning methods to discover the Supersymmetry particles. While the third section will describe the "Susy" dataset. The four section will recap our experimental and their consequent analysis and the last section summarizes the conclusions extracted from experimentation and analysis steps.

II. PROPOSED METHODS

A. Spark Environment

Apache Spark is an open source and ecosystem designed to Big Data. This includes a common Machine Learning library (MLlib) with many languages as Java, Scala, python, SQL and R environment works well with python. Spark works well in memory with many languages and handles successfully Big data as such "Susy" dataset with the Pyspark programs [7,8,9]. In this document, we use Spark-Hadoop 2.6.4 with the Jupyter cloud to training and testing the Machine Learning methods via the Pyspark environment.

B. Proposed Machine Learning methods

This section presents brief definitions of different Machine Learning classifiers that we used to resolve the problem of supersymmetry particles classification:

- Logistic Regression (LR) Method:

The LR is a classification method with powerful predictors. LR aims to predict or explain a categorical response with two modalities (positive, negative) depending on predictors. It focuses only on the ratio of the probabilities.
In practice, we use the spark.ml logistic regression module to predict a binary outcome [10].

- Decision Tree (DT) Method:

The DT method is a weak learner used to solving regression or classification problems [11]. It contains nodes, branches, and leaves: Node designs an attribute, branch represents a decision rule and leaf represents an outcome (see Fig.1). We can build a decision tree for classification problems using many algorithms as ID3 and CART [12].

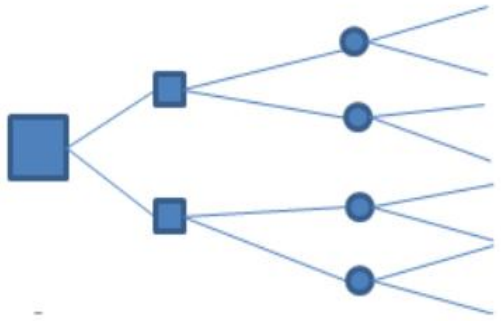

Fig. 1.DecisionTreegraph-source:https://www.manager-go.com/

- Random Forest (RF) Method:

The RF is a learning method that consists of multiple decision trees in order to build the final decision using the majority voting [13]. The principal benefit of RF is that it reduces the risk of over-fitting, offers a high level of accuracy and runs efficiently in large databases [14]. The bellow figure explains how works Random Forest algorithm:

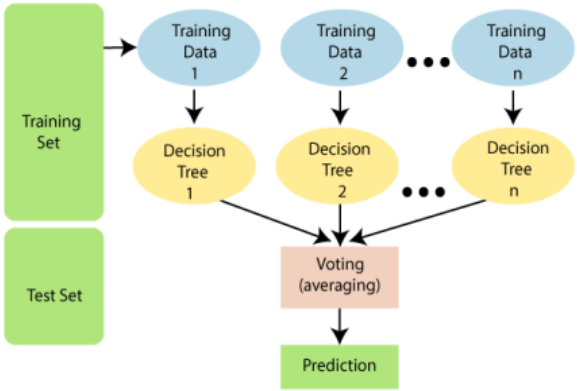

Fig.2. Simplified Random Forest- Source:https:// www. javatpoint.com/

- Gradient Boosted Tree (GBT) Method:

The Boosting method consists of converting weak learners into strong learners minimizing a loss function. The principal idea of Boosting is to join a new classifier to the ensemble sequentially. Gradient Boosted tree (GBT) is an ensemble method that uses an ensemble of decision trees to predict a target label. The spark.ml implementation supports GBTs for binary classification [15].

III. "SUSY" DATASET DESCRIPTION

A. Features and Distribution

In this work, we use the "Susy" dataset (Supersymmetric particles), downloaded from UCI site. We use eighteen (18) features as independent variables and one dependent variable with two classes (signal and background events) [16]:

- Eight (8) Low-Enegy input features:

F1= "lepton1_pt", F2=" lepton1_eta", F3= "lepton1_phi", F4= "lepton2_pT", F5= "lepton2_eta", F6= "lepton2_ phi", F7= "missing energy magnitude", F8= "missing energy phi".

- Ten (10) High-Energy input features:

F9= "MET_rel" , F10= "axial MET", F11= "M_ R"" , F12= M_TR2", F13= "R" , F14="MT2", F15= "S_R", F16= "M_Delta_ R", F17= "dPhi_r_ b ", F18=" cos_theta_r1".
The class (signal, background) is response variable. It consists of 54\% (2 712 173 examples) of the signal and 46\% of background events (2 287 827 examples). We conclude that the distribution of the "Susy" dataset is nearly balanced: signal=46% and background= 54% (see Table -I)

Table-I: Susy dataset description: instances, features, class label and split data.

Designation	*Description*	
Instances	*5000000*	
Class label (signal, background)	*(s=1;b=0)*	*(1;0)*
Signal	*2 712 173*	*54%*
Background	*2 287 827*	*46%*
Training set	*3 500 000*	*70%*
Test set	*1 500 000*	*30%*

B. Split Data

In the aim to get an unbiased estimation of the algorithm performances of the, "Susy" dataset is divided into a training sample (70%) and test sample (30%).

C. Evaluation Functions

In the evaluation phase, we use three metrics: Accuracy, Area Under Curve (AUC) and training Computation Time (CT):

- Accuracy metric:

Accuracy is defined as the proportion of total instances that are correctly classified [17]. The following equation measures this metric:

$$Accuracy = \frac{true\ positive + true\ negative}{true\ positive + true\ negative + false\ positive + false\ negative} \quad (1)$$

Where:
The true positive is the total of instances that are "Susy" particles and are being classified as "Susy" particles;
The true negative is the total of instances which are no-"Susy" particles and being classified as no-"Susy" particles;
The false negative is the total of true "Susy" particles that are wrongly being classified as no-"Susy" particles;
The false positive is the total of no-"Susy" particles that are wrongly being classified as "Susy" particles.

- ROCC-AUC metric:

The Receiver Operating Characteristic Curve (ROCC) is a metric that assesses the model's ability to distinguish binary classes. The ROCC allows comparing various supervised learning classifiers. It is especially useful for cases of skewed class distribution [18].

The Area Under Curve (AUC) is an area equivalent to the probability that the algorithm will classify a random positive instance over a random negative instance [19].

- Computation Time (CT) metric:

The CT is the time elapsed by an algorithm during the training phase. In this work, all Algorithms were trained using Pyspark programs with the following features:

- Intel (R) cores (TM) i7;
- 7500U CPU@ 2.7 GHz 2.9 GHz;
- 8 Go memory (RAM);
- processor x64.

IV. EXPERIMENTATION, RESULTS AND ANALYSIS

A. Experimentation Step by Step

In this subsection, we present the different steps followed using code program with pyspark and MLlib library:

- Connect to spark

```
#Connect to spark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('').getOrCreate()
# install sql packages
import pyspark
from pyspark.sql.types import *
from pyspark.sql import Row
from pyspark.sql.functions import UserDefinedFunction
```

- Installation of Machine learning method packages

```
##Logistic regreesion
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer
from pyspark.mllib.classification import LogisticRegressionWithLBFGS
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.regression import LabeledPoint
```

```
##decision tree
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
##Random forest
from pyspark.mllib.tree import RandomForest
from pyspark.ml.classification import RandomForestClassifier
##gradient boosted tree
from pyspark.ml.classification import GBTClassifier
```

```
##SparseVector
from pyspark.mllib.linalg import SparseVector
## Hashing and vectorization
from pyspark.ml.feature import HashingTF
from pyspark.ml.feature import IDF
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

- Download Dataset Susy.csv

```
#download dataset
dataset = spark.read.csv("d://data_higgs\\susy.csv",
                         header=True,
                         inferSchema=True)
dataset.printSchema()
```

- Preprocessing dataset

```
#preprocessing data
assembler=VectorAssembler(inputCols=['lepton1_pT','lepton1_eta',
                                     'lepton1_phi','lepton2_pT',
                                     'lepton2_eta','lepton2_phi',
                                     'missing_energy_magnitude',
                                     'missing_energy_phi',
                                     'MET_rel', 'axial_MET', 'M_R',
                                     'M_TR_2','R','MT2','S_R',
                                     'M_Delta_R','dPhi_r_b',
                                     'cos_theta_r1'

 ],outputCol='features')
output_data=assembler.transform(dataset)
final_data=output_data.select('features','classLabel')
```

- Split data into 70% training and 30% testing

```
#split data
train,test=final_data.randomSplit([0.7,0.3])
```

- Logistic Regression: training and testing steps

```
#logistic regression
##training step logistic regression
model=LogisticRegression(labelCol='classLabel')
from time import time
t0 = time()
model=model.fit(train)
tt = time() - t0
print("Classifier trained in {} seconds".
      format(round(tt,3)))
 ##testing step logistic regression
evaluator=BinaryClassificationEvaluator(rawPredictionCol='prediction',
                                        labelCol='predictions')
```

- Decision tree: training and testing steps

```
# decision tree
##  training decision tree step
from time import time
t0 = time()
dt = DecisionTreeClassifier(featuresCol = 'features',
                            labelCol = 'classLabel',
                            maxDepth = 3)
dtModel = dt.fit(train)
tt = time() - t0
print("Classifier trained in {} seconds".format(round(tt,3)))
##  tetsing step
predictions = dtModel.transform(test)
predictions.select('classLabel', 'rawPrediction',
                   'prediction', 'probability').show(10)
```

- Random Forest: training and testing steps

```
# Random Forest
##  step training Random Forest
rf = RandomForestClassifier(numTrees=20, maxDepth=5,
                            featuresCol = 'features',
                            labelCol = 'classLabel')
from time import time
t0 = time()
rfModel= rf.fit(train)
tt = time() - t0
print("Classifier trained in {} seconds".format(round(tt,3)))
## test step Random Forest
predictions = rfModel.transform(test)
predictions.select('classLabel', 'rawPrediction', 'prediction',
                   'probability').show(10)
```

- Gradient Boosted trees: training and testing steps

```
# gradient boosted tree
## training step gbt
gbt = GBTClassifier(featuresCol = 'features',
                    labelCol = 'classLabel')
from time import time
t0 = time()
gbtModel = gbt.fit(train)
tt = time() - t0
print("Classifier trained in {} seconds".format(round(tt,3)))
##testing step gbt
predictions = gbtModel.transform(test)
predictions.select('classLabel', 'rawPrediction',
                   'prediction', 'probability').show(10)
```

- Accuracy metric

```
##accuracy metric
evaluator = MulticlassClassificationEvaluator(
labelCol="classLabel", predictionCol="prediction",
    metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test Error = %g" % (1.0 - accuracy))
print(accuracy)
```

- ROCC-AUC metric

```
##Rocc- AUC metric
from pyspark.ml.evaluation
import BinaryClassificationEvaluator
evaluator=BinaryClassificationEvaluator(rawPredictionCol=
                  'prediction',labelCol='classLabel')
print("Test Area Under ROC:
      " + str(evaluator.evaluate(predictions,
      {evaluator.metricName: "areaUnderROC"})))
```

B. Results presentation

The table-II, table-III, and table -IV Present respectively the performance of low-level input features, high-level input features, and all input features:

Table-II. Performance of Low-Energy input features

Classifier	*Accuracy*	*AUC*	*CT (s)*
Logistic Regression (LR)	*0,768*	*0,8312*	*89,869*
Random Forest (RF)	*0,775*	*0,7697*	*518,11*
Decision Tree (DT)	*0,7588*	*0,7474*	*219,173*
Gradient Boosted Tree (GBT)	*0,7816*	*0,7746*	*2358,773*

Table-III. Performance of High-Energy input features

Classifier	*Accuracy*	*AUC*	*CT (s)*
Logistic Regression (LR)	*0,7702*	*0,8336*	*254,7020*
Random Forest (RF)	*0,7621*	*0,7528*	*467,4970*
Decision Tree (DT)	*0,7485*	*0,7374*	*249,6220*
Gradient Boosted Tree (GBT)	*0,7827*	*0,7748*	*4 907,0490*

Table-IV. Performance of all input features: Low-Energy and Hight-Energy

Classifier	*Accuracy*	*AUC*	*CT (s)*
Logistic Regression (LR)	*0,7884*	*0,8576*	*2148,734*
Random Forest (RF)	*0,774*	*0,7658*	*564,944*
Decision Tree (DT)	*0,7546*	*0,7416*	*627,392*
Gradient Boosted Tree (GBT)	*0,793*	*0,7862*	*5468,769*

C. Analysis and Discussion

In this part, we compare different models on "Susy" dataset in terms of AUC, Accuracy and Computation Time metrics.

In the context of Low-Energy input features, the accuracy metric varies between 76% for the DT algorithm and 78% for the GBT model (See Fig.3). However, the LR method achieved a higher AUC score with 83\% better than GBT and RF with 77%, that better than DT with 75% (See Fig.4)

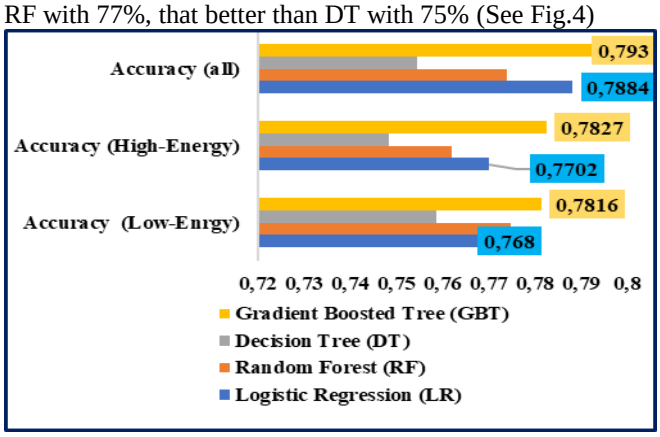

Fig.3. Accuracy of Low-Energy, High-Enery, and all input features

On the other hand, Fig.5 shows that RL classifier runs quickly (89 seconds), whereas the GBT method trained very slowly in 2358 seconds. In the case of High-Energy, similarly, the accuracy metric varies between 75% for the DT algorithm and 78% for the GBT model. However, the LR method achieved a higher AUC score with 83% better than GBT with 77% and RF with 75%, that better than DT with 74% (See Fig.4). On the other hand, the Fig.5 shows that DT classifier, LR, and RF algorithms run quickly, whereas the GBT method were trained very slowly in 4907 seconds.

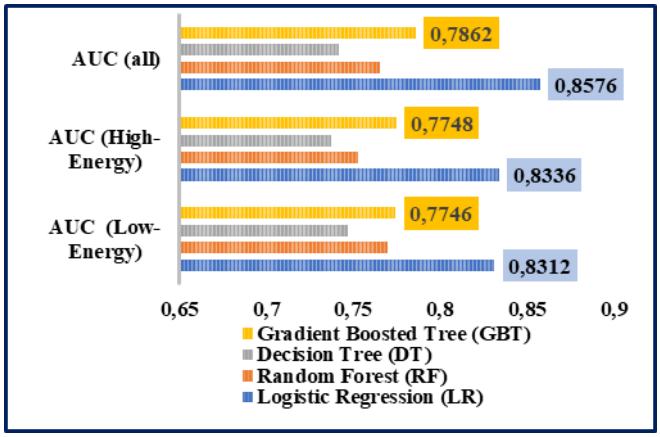

Fig.4. AUC of Low-Energy, High-Enery, and all input features

As part of the complete input features (Low-Energy and High-Energy), the accuracy metric is improved. The score varies between 75% for the DT algorithm and 79% for the GBT model.

However, the LR method achieved a higher AUC score with 86% better than GBT with 79% and RF with 76%, that better than DT with 74%. On the other hand, Fig.5 shows that DT classifier and RF algorithm run quickly, whereas the GBT method trained very slowly in 5468 seconds.

Those results are closes to the score of Higgs challenge 2014 (83%) [1] and are very close to those obtained using the Deep Learning method [3,4,5].

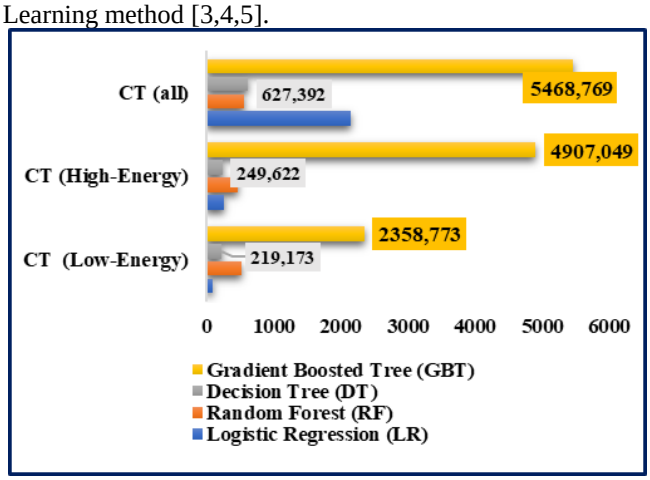

Fig.5. Computation Time of Low-Energy, High-Enery, and all input features

V. CONCLUSION

In this paper, we attempted to apply a very powerful data analysis technique: Pyspark environment with the MLlib library. The Pyspark ecosystem has been implemented to the "Supersymmetric Particles" dataset using four Machine Learning methods. Then, three contexts of energy particles were explored: Low-Energy features, High-energy features, and all Energy features.

In the area of Exotic Particle Discovery, we conclude that:

Concerning the Low-Energy features, The Gradient Boosted Tree (GBT) classifier runs well with accuracy achieved 78% while LR works efficiently with AUC (83%).

As part of the High-Energy feaures, we remark a similar conclusion, the performance of the Gradient Boosted Tree (GBT) classifier is good with accuracy 78%. Whereas, Logistic Regression outperforms the AUC score of (83%).

In the condition of the complete input features, The Gradient Boosted Tree (GBT) classifier achieves a high accuracy score (79%), while the LR exceeds other classifiers with 86%.

Concerning the Computation Time metric, the GBT method works very slowly, particularly in the case of the complete input features.

REFERENCES

1. C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kegl, and D. Rousseau,"The Higgs Machine Learning Challenge, Journal of Physics: Conference Series", vol 634, (2015), pp. 072015. doi:10.1088/1742-6596/664/7/072015
2. T. Chen and T. He, “Higgs Boson Discovery with Boosted Trees, JMLR: Workshop and Conference Proceedings, vol 42, (2015), pp.69-80.
3. P. Baldi, K. Cranmer, T. Faucett, P. Sadowski, and D. Whiteson “Parameterized Machine Learning for High-Energy Physics”,The European Physical Journal, vol 76, (2016), pp. 235-241. doi.org/10.1140/epjc/s10052-016-4099-4.
4. P. J. Sadowski, D. Whiteson, and P. Baldi, “Searching for Higgs Boson Decay Modes with Deep Learning”, Advances in Neural Information Processing Systems (NIPS), vol 27, (2014)
5. P. Sadowski, J. Collado, D. Whiteson, and P. Baldi, “Deep Learning, Dark Knowledge, and Dark Matter.”, JMLR: Workshop and Conference Proceeding, vol 42, (2015), pp.81-97.
6. A. Alves, “Stacking machine learning classifiers to identify Higgs bosons at the LHC”, Journal of Instrumentation, vol 12, (2017), pp. T05005-T05005. doi.org/10.1088/17480221/12/05/T05005.
7. X. Meng et al, "MLlib: Machine Learning in Apache Spark", Journal of Machine Learning Research, Boston, MA, vol 17, 2016), pp.1-7.
8. M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, “Big data machine learning using apache spark MLlib”, in 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, (2017), pp.3492–3498. doi: 10.1109/BigData.2017.8258338
9. M. Armbrust et al., “Spark SQL: Relational Data Processing in Spark”, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data -SIGMOD '15, Melbourne, Victoria, Australi, (2015), pp.1383–-1394. doi: 10.1145/2723372.2742797.
10. H. Peng, D. Liang, and C. Choi, “Evaluating parallel logistic regression models”, IEEE International Conference on Big Data, Silicon Valley, CA, USA, (2013), pp.119-126. doi: 10.1109/BigData.2013.6691743.
11. W. Loh,"Classification and regression trees", Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol 1, (2011), pp.14-23. doi: 10.1002/widm.8.
12. R. Genuer and J.-M. Poggi, « Arbres CART et forets aléatoires-Importance et sélection de variables », HAL Id: hal-01387654, (2017). https: hal.archives-ouvertes.fr/hal 01387654v2.
13. B.Leo, “Random Forests”, Machine Learnin, vol 45, (2001), pp.5-32.
14. A. Liaw and M. Wiener, “classification and regression by random fores”, R News, vol 2, (2002), pp.18-22.
15. G.Yasser. , C. Rich , and L. Cristina Videira, “Bagging gradient-boosted trees for high precision, low variance ranking models.”, In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, Beijing, China, (2011), pp.85-94. doi: 10.1145/2009916.2009932.
16. P.Baldi, P.Sadowski, and D. Whiteson, “Searching for Exotic Particles in High-energy Physics with Deep Learning.”, Nature Communications, vol 5, (2014). doi.org/10.1038/ncomms5308.
17. R. G. Congalton, “A Review of Assessing the Accuracy of Classifications of Remotely Sensed Data.", Remote Sensing of Environment, vol 37, (1991), pp.35–46. doi.org/10.1016/0034-4257(91)90048-B.
18. R. Kumar and A. Indrayan, “Receiver operating characteristic (ROC) curve for medical researchers, Indian Pediatrics, vol 48, (2011), pp.277–287. doi: 10.1007/s13312-011-0055-4.

19. T. Fawcett, “An introduction to ROC analysis”, Pattern Recognit, vol 27, (2006), pp. 861-874.
doi: 10.1016/j.patrec.2005.10.010.

AUTHORS PROFILE

Mourad Azhari is a Ph.D at the Faculty of Sciences, Kenitra, Ibn Tofail University, Morocco. He obtained an engineering degree in operational research at the National Institute of Statistics and Applied Economics of Rabat in 2013, Morocco. His research interests cover the study of Machine Learning methods in the context of Big Data using the datasets in the physic area.

Abdallah Abarda has experience of over ten years in the field of data science, applied statistics, survey methodology, and artificial intelligence. Data scientist professor with advanced statistical skills and experience programming with Python and R software

Badia Ettaki is a professor at the Engineering School of Information Sciences in Rabat, Morocco. She obtained a doctoral degree in the Statistic Physic. His research interests cover the study of Content and Knowledge Engineering, Machine Learning algorithms and Big Data analytics.

Jamal Zerouaoui is a professor at the Faculty of Sciences, Kenitra, Ibn Tofail University, Morocco. He received a doctoral degree in the physic science. Currently, his research interests cover modeling usinMachine Learning methods in the physic area.

Mohamed Dakkon is got a doctoral degree in Statistics He is a professor at Abdel Malek Essaadi University, Tetouan, Morocco. His research interests include the performance of Machine Learning, Data Analysis, Classification, and Big Data.