Fake Item Machine Learning Classifiers

Gustavo V. N. Luizon, Tiago E. P. C. Conceição

[1]Coimbra University, Coimbra, Portugal

Abstract. Considering the importance of increasing security against falsifications that are increasingly present in digital media, this paper aims to evaluate the performance of several machine learning methods capable of evaluating the verity of a set of images.

Keywords: Deepfake · Machine Learning · Data Science · Fake · RFC · SVM · MLP · KNN · GNB

1. Introduction

Falsification has been a matter of concern from the birth of the mercantilist world to its evolution to capitalist production systems we have today, however with the evolution of digital resources, more advanced and alarming falsification methods emerge, such as "deepfake", in which a person in an existing image or video is replaced with someone else's likeness.
Deepfakes can bring consequences that go far beyond financial damage, the primary threat of deepfakes stems from people becoming convinced that something fictional really occurred. But deepfakes can distort the truth as manipulated videos pervade the internet, it may become progressively harder to separate fact from fiction.
In order to take the first steps towards identifying and preventing falsification, this paper evaluates the performance of five different machine learning classification methods to identify images of fake items. Here is a list of evaluation methods:

1.1. Random Forest Classifier – RFC

1.2. Support Vector Machines Classifier – SVM

1.3. Multi-layer Perceptron Classifier – MLP

1.4. K Neighbors Classifier – KNN

1.5. Gaussian Naïve Bayes Classifier – GNB

2. Dataset

The dataset, for this project is composed by 40k image files divided into two groups, 20k files with fake item images and 20k files with real item images, this dataset was used to train and evaluate the models. Another dataset, with other 2000 image files is available for testing and evaluation in the "DEI UC Advanced Machine Learning Competition".
The images are composed by 28*28 pixels in grayscale and each one of those represents one requirement.

3. Pre-Processing

The machine learning algorithms used to classify the items are provided by the "Scikit Learn" library developed for the python programming language.
The Scikit Learn library requires that each input data be structured in a matrix composed of a row for each item and a column for each feature. As the dataset consists of 28 x 28 pixels images, the data has been resized to 1x 784 arrays to comply with requirements.
The complete training dataset after resizing had dimensions of 40000 x 784, this dataset was then divided into a training dataset and a test dataset with labels so that it is possible to evaluate the machine learning algorithms before their effective application to the test dataset of 2k elements. (1. Scikit Learn, s.d.)

4. Methodology

In this chapter contains the methodology description of each algorithm used in this project and the way that those algorithms were used.
The problem consists in a binary classification, each algorithm was evaluated in different parameter configuration according to its specificities.
In order to measure the performance of which algorithm, the following metrics were applied:

- Accuracy: computes the set of labels predicted for a sample must exactly match the corresponding set of labels;
- Precision: The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives;
- Recall: The recall is intuitively the ability of the classifier to find all the positive samples. The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives;
- Roc Auc: Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

4.1. Random Forest Classifier

The Random Forest Classifier is a machine learning classifier algorithm which belongs into the supervised learning techniques. The Random Forest classifier fits a number of decision tree classifiers on various sub-samples and uses the average of results obtained to improve the accuracy of the predictions when compared to results obtained using a single tree.
Some advantages that this process as, is that it's capable to perform both, classification and regression, it presents good performance when handling large datasets.

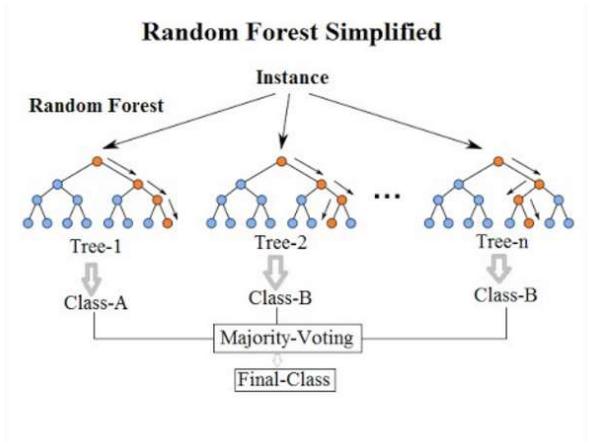

Figure 1- Random Forest Classifier

The most important parameters to be defined when using the algorithm are described as follows:

- N_estimators: defines the number of trees in the forest
- Max_depth: The depth of the tree, expand all nodes until all leaves are pure
- Criterion: The function to measure the quality of a split.

(2. Random Forest Classifier, s.d.)

4.2. Support Vector Machines Classifier

The Support Vector Machines Classifier algorithm is a supervised learning technique that can be used for classification and regression problems. In this type of algorithm, the objective is to set the decision boundary dividing this way the data into classes, so that in the future the new data is set in the correct class.
The boundary that divides the data is called hyperplane. The library that supports this algorithm is scikit learn, and has three different kernels: Linear, RBF, Poly.

Each one of these kernels have a purpose, the first one is dedicated to when the data is linearly separated, which is not the case of our project, that's why the other two are perfect for the context, because they are not used for data not linearly separable.

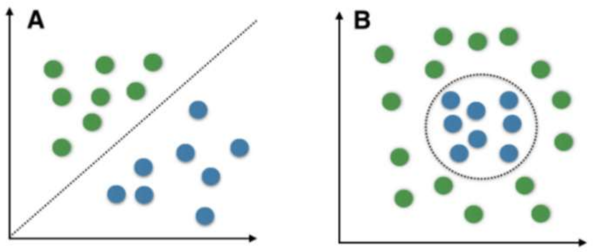

Figure 2 - Support Vector Machines Classifier

The most important parameters to be defined when using the algorithm are described as follows:

- C: Parameter trades off correct classification of training examples against maximization of the decision function's margin, higher C smaller margin accepted.
- Gama: Parameter that defines how far the influence of a single training example reaches, with low values we get further and with high values we get closer, and our purpose.

- Kernel: Mathematical function used to dimensional transforming of data
- Degree: Controls the flexibility of the decision boundary. Higher degree a more flexible boundary.

(3. Support Vector Machines Classifier, s.d.)

4.3. Multi-layer Perceptron Classifier

The Multi-Layer Perceptron Classifier is a neural network that maps between inputs and outputs, and hidden layers with many neurons stacked together. The MLP is considered a feedforward artificial neural network algorithm, the multi-layer perceptron data flows in the forward direction from input to output layer. The neurons in the MLP algorithm are trained with back propagation learning algorithm, these are designed to solve problems which are not linearly separable, like the ones mentioned before this procedure is indicated to pattern classification, recognition, and prediction.

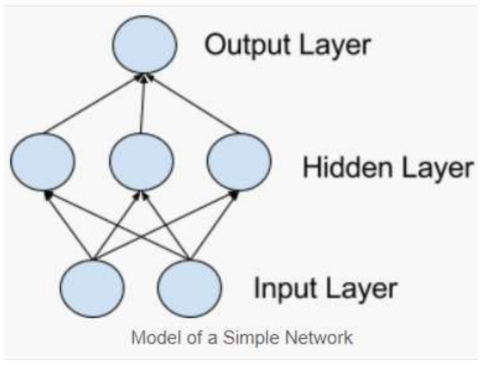

Figure 3 - Multi-layer Perceptron Classifier

The most important parameters to be defined when using the algorithm are described as follows:

- Solver: The solver for weight optimization, we used 'adam';
- Max_Iter: Determines how many times each data point will be used;
- Alpha: Strength of the L2 regularization term;
- Hidden Layer Size: Perform nonlinear transformations of the inputs entered the network;
- Random State: Determines random number generation for weights and bias initialization.

(4. Multi-Layer Perceptron Classifier, s.d.)

4.4. K Neighbors Classifier

The Neighbors Classifier Algorithms Class can be used to solve unsupervised and supervised neighbors-based learning methods, the principle behind of these methods is to find a predefined number of training samples closest in distance to the new point and predict the label from these.
The supervised neighbors-based learning comes in classification for data with discrete labels and regression for data with continuous labels, for the problem presented in this paper, the adequate method is K Neighbors Classifier algorithm, it implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. Higher values of k suppress the effect of noise but makes the classification boundaries less distinct.

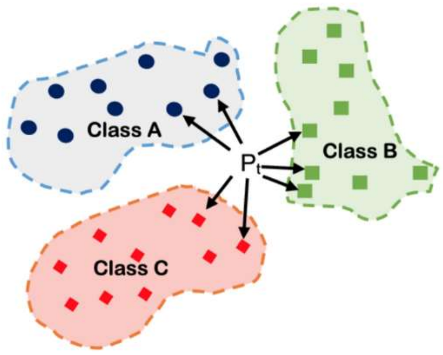

Figure 4 - K Neighbors Classifier

The most important parameters to be defined when using the algorithm are described as follows:

- N_neighbors: Number of neighbors to use by default for neighbor's queries
- Weights: "uniform" weights option means that all points in each neighborhood are weighted equally. The 'distance' option weights points by the inverse of their distance, in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away
- Algorithm: Algorithm used to compute the nearest neighbors: 'auto' will attempt to decide the most appropriate algorithm based on the values passed to fit method.
- P: Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2.

(5. K Neighbors Classifier, s.d.)

4.5. Gaussian Naive Bayes Classifier

A Naive Bayes classifier is a probabilistic classifier based on applying Bayes theorem with strong independence assumptions. In other words, a naive Bayes classifier assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of any other feature. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all these properties to independently contribute to the probability.
The Naïve Bayes parameters are maximum likelihood estimates of the probabilities and can be approximated with relative frequencies from the training set.

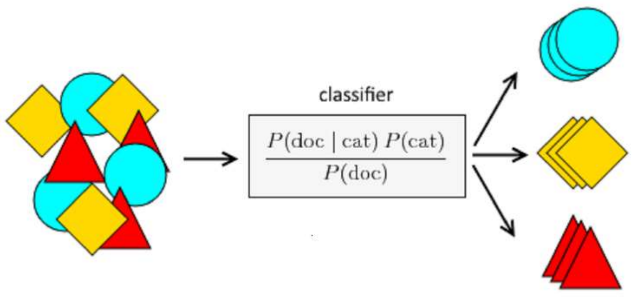

Figure 5 - Gaussian Naive Bayes Classifier

The algorithm does not use parameters, but allows the choice of different probability distributions for the features:

- Feature likelihood: Gaussian, Multinomial, Complement, Bernoulli, Categorical

(6. Gaussian Naive Bayes Classifier, s.d.)

5. Results

The results obtained by each algorithm and the corresponding metrics, accuracy will show how precise the value of our classification is, the precision metric is defined by the quality of the value being accurate, recall metrics, that is the portion of the relevant data that can be retrieved and finally the “roc auc” metric, that defines the performance of a classification model.
Each one of the algorithms have been through some extensive lookup of hyper parameters in order to obtain the best results possible. The Scikit Learn inspection techniques used to search and get the best metrics configuration are called “GridSearchCV” and “HalvingGridSearchCV”, the methods allow the user to previously choose the set of parameters which will be applied in the training process, and the techniques will execute every combination of them for the chosen algorithm and return the evaluation through the specified metrics.
The Grid Search techniques demands high computational power and can require significant processing times when combined with large datasets and highly complex algorithms. In order to avoid impractical processing times, previous sensitivity tests were performed to reduce the size of the simulated parameter set.

5.1. Random Forest Classifier

The set of parameters used to obtain the results through the Random Forest Classifier is specified in table 1.

Table 1 - Random Forest Classifier Parameters

Parameter	Values
N_estimators	500; 1000; 3000
Max_depth	50; 150; 300
Criterion	Gini; Entropy

The Random Forest Classifier obtained results shows high rate values for all applied metrics, figure 6 presents five exemplary results of the combinations of parameters used, the other configurations were omitted since they present identical or very similar performance to these.
All parameter combinations showed very similar results, however, a small efficiency gain for the accuracy metric is observed when using the "Entropy" type for the criterion parameters. In addition, efficiency gains were also found for higher values of the "n_estimators" parameter, as expected.

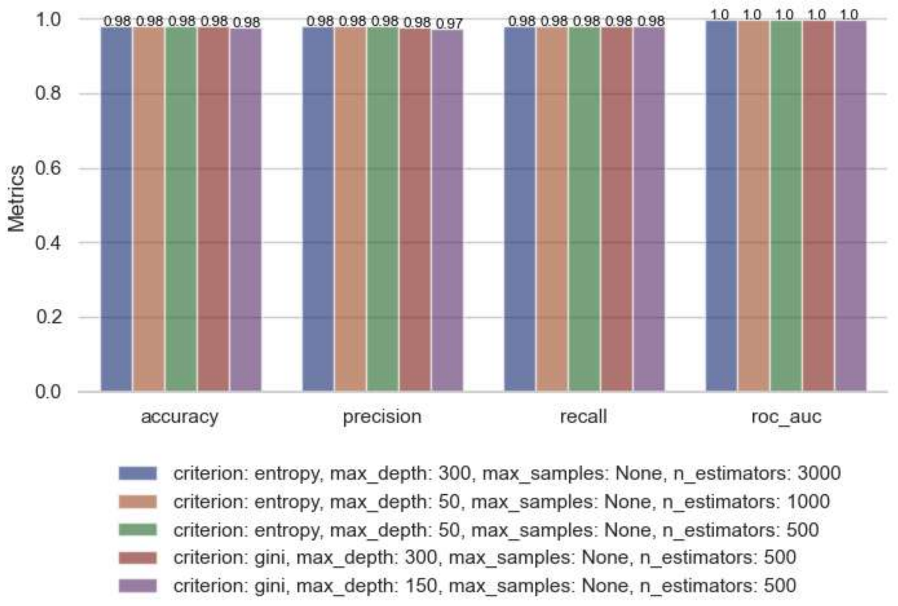

Figure 6 - Random Forest Classifier Results

5.2. Support Vector Machines Classifier

The set of parameters used to obtain the results through the Support Vector Machines Classifier is specified in table 2.

Table 2 - Support Vector Machines Classifier Parameters

Parameter	Values
C	0.1;10;100
Gama	0.01;0.1;1
Kernel	Poly; Rbf
Degree	2;3;4

The Support Vector Machines Classifier obtained results shows high rate values for all applied metrics, the figure 7 presents five exemplary results of the combinations of parameters used, the other configurations were omitted since they present identical or very similar performance to these.
All parameter combinations showed rate results varying between 0.9 and 0.99, which means a high performance, it has been verified a loss in the accuracy metric when using higher values for de "degree" param.

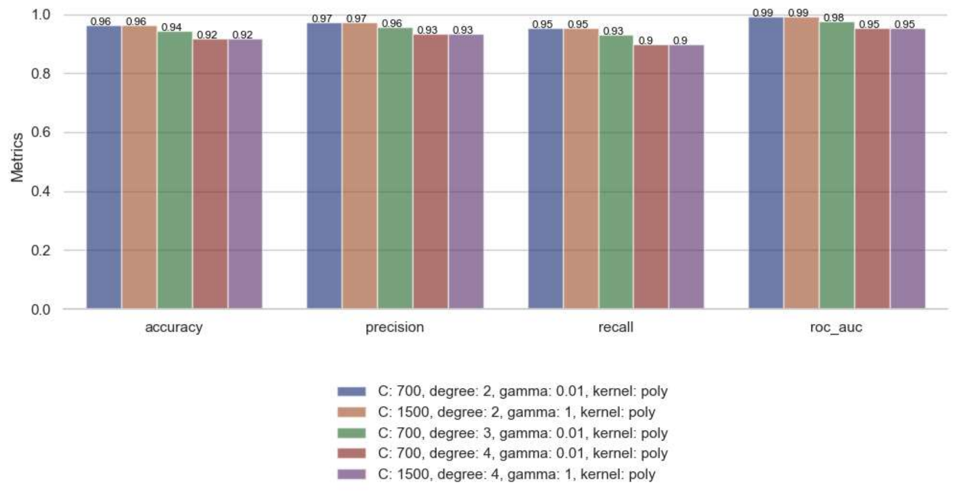

Figure 7 - Support Vector Machines Classifier Results

5.3. Multi-layer Perceptron Classifier

The set of parameters used to obtain the results through the Multi-layer Perceptron Classifier is specified in table 3.

Table 3 - Multi-layer Perceptron Classifier Parameters

Parameter	Values
Solver	Adam
Max_Iter	1000;1500;2000
Alpha	0.1;0.05;1
Hidden Layer Size	200;500
Random State	1;3;5

For the solver parameter ‘adam’ was selected considering its adequate performance to large dataset, getting the best training time and validation score.
The Multi-layer Perceptron Classifier obtained results shows high rate values for all applied metrics, the figure 8 presents five exemplary results of the combinations of parameters used, the other configurations were omitted since they present identical or very similar performance to these.
All parameter combinations showed metric rate results varying between 0.84 and 0.98, which means a high performance. Among the sets of parameters tested, it was not possible to determine the particular predominance of any parameter over the results.

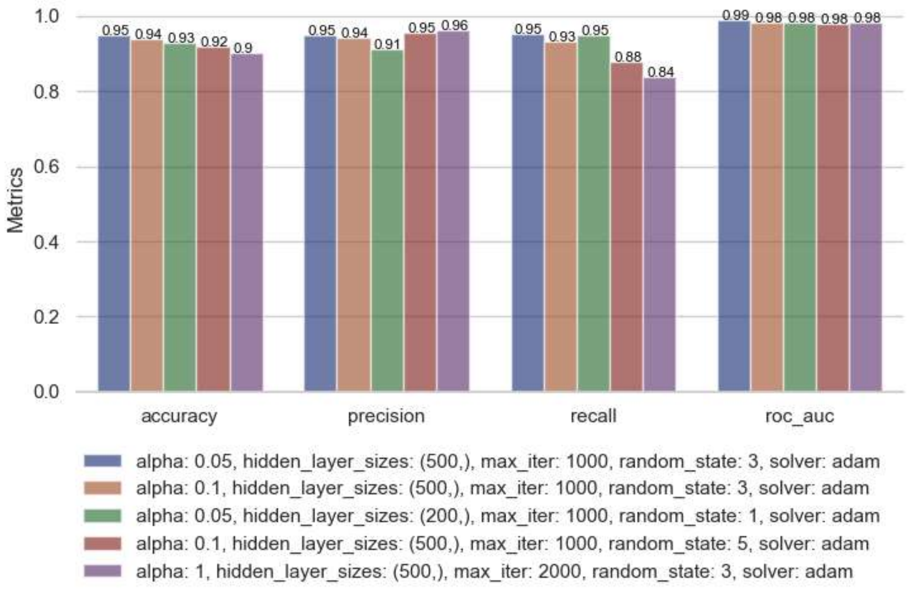

Figure 8 - Multi-layer Perceptron Classifier Results

5.4. K Neighbors Classifier

The set of parameters used to obtain the results through the K Neighbors Classifier is specified in table 4.

Table 4 - K Neighbors Classifier Parameters

Parameter	Values
N_neighbors	1;3;5
Weights	Uniform; Distance
Algorithm	Auto; kd_tree
P	1:2

The K Neighbors Classifier obtained results show high rate values for all applied metrics, the figure 9 presents five exemplary results of the combinations of parameters used, the other configurations were omitted since they present identical or very similar performance to these.
All parameter combinations showed rate results varying between 0.76 and 0.95, which means a high performance, it has been verified a loss in the accuracy metric when using Euclidean distance for “p” param.

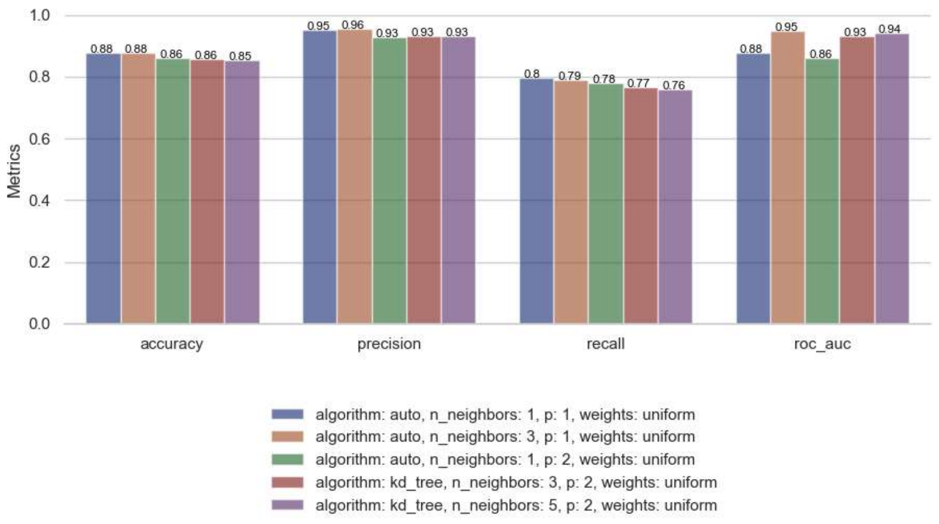

Figure 9 - K Neighbors Classifier Results

5.5. Gaussian Naive Bayes Classifier

For the Gaussian Naïve Bayes Classifier, the "var_smoothing" parameter has the purpose to watch the portion of the largest variance of all features that is added to variances for calculation stability.
The predict simulations showed metric rate results varying between 0.54 and 0.58 for accuracy and low values for the recall metric, the precision and "roc_auc" metrics present results between 0.65 to 0.69, which means a regular performance for the Gaussian Naïve Bayes Classifier as shown at figure 10.

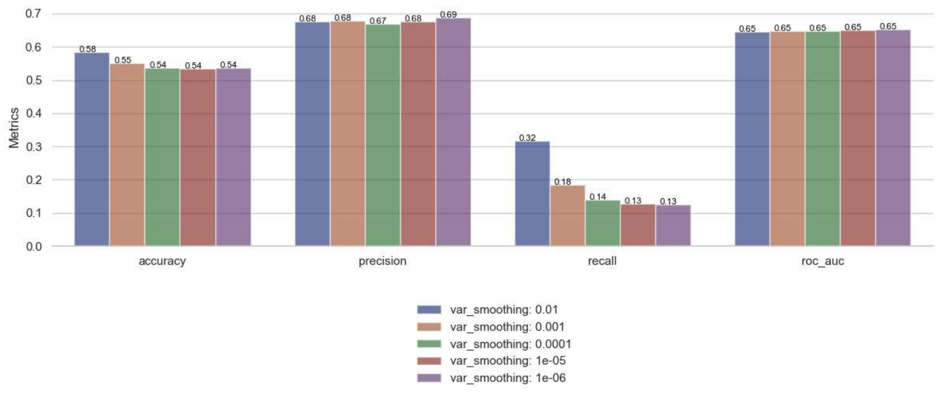

Figure 10 - Gaussian Naive Bayes Classifier Results

6. Conclusion

The presented paper proposes a comparative evaluation between different methods and configurations for the detection of fake items through an image evaluation. The results obtained prove that it is not possible to define a single approach as the ideal one for the solution of this problem, each method has its positive and negative points, presents performance variation according to the parameter configuration presented and has different requirements for computational processing.
When compared to each other, the results show a superior performance for the random forest, SVM and MLP algorithms, followed by the KNN algorithm that presents slightly inferior results and with a much lower evaluation than the others, we have the GNB algorithm.
Despite presenting an approximately equivalent performance between the RFC, SVM and MLP algorithms, it is important to notice that the MLP algorithm presents a higher computational demand and consequently requires a much higher processing time than the other two.
Furthermore, the considerable low performance of the GNB method when compared to the other evaluated methods shows that it is not an interesting approach to solve this type of problem.

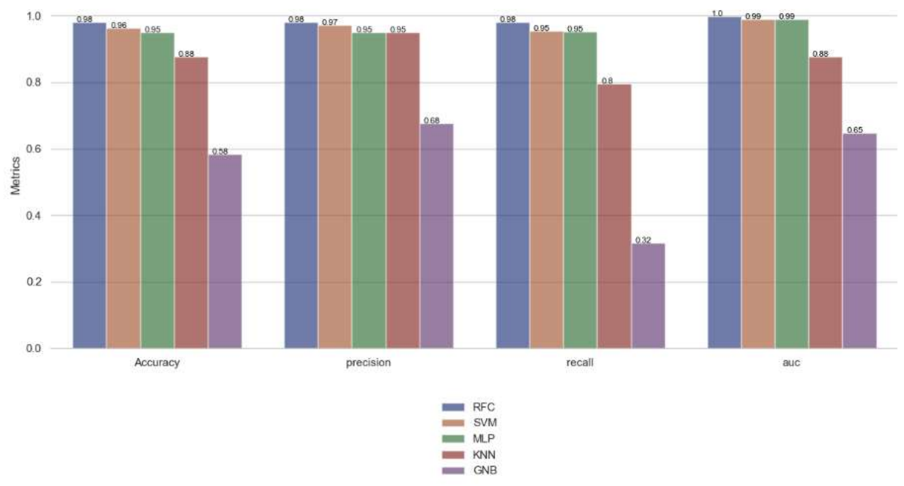

Figure 11 - Comparative Methods Performance

7 Referencies

1. *Scikit Learn*. (s.d.). Obtido de https://scikit-learn.org/stable/
2. *Random Forest Classifier*. (s.d.). Obtido de https://machinelearningmastery.com/types-of-classification-in-machine-learning/.
3. *Support Vector Machines Classifier*. (s.d.). Obtido de https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
4. *Multi-Layer Perceptron Classifier*. (s.d.). Obtido de https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron
5. *K Neighbors Classifier*. (s.d.). Obtido de https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
6. *Gaussian Naive Bayes Classifier*. (s.d.). Obtido de https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148