Random Forest for Loan Approval

Arslonbek Ishanov

Abstract. Many industries try to keep pace with the technologies, and banks are no exception. One of them is Artificial Intelligence. It plays a crucial role in attracting new customers, answering their inquiries, and even deciding whether their loan application would be approved. However, over the past decades, various models of Artificial Intelligence have been developed. One of them is Random Forest. This report will analyse the algorithm behind Random Forest, what are its advantages, how can it be improved, and what potential implications it might have for loan applicants and banks.

1 Introduction

Since its invention in the 1950s, Artificial Intelligence, commonly known as AI, found a plethora of use cases in the daily lives of almost all people. From home assistants such as Siri and Alexa to autonomous vehicles, and from recommendation systems on social media to fraud detection. However, it is worth noting that there is not a definite answer to what AI actually is. Oxford Languages has the following definition of AI: "Computer Systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages" [11], while technological giants such as Google and IBM have entire webpages dedicated to explaining what AI is[3][9]. Nowadays, more and more organisations show interest in implementing AI into their processes. And financial institutions are no exceptions. One of the primary objectives of financial institutions is lending money. At the moment, Artificial Intelligence has been implemented in all major stages of this process. Forbes demonstrates this in its recent article called "Artificial Intelligence Across The Lending Life Cycle"[6]. Moreover, LinkedIn published an article that discovered significant benefits for all participants in such a trend. These advantages include more accurate digital credit history analysis, lower costs for clients, enhanced risk management, and reduced bias[10]. However, not all Artificial Intelligence is created equal as there are different models. Most popular are, but not limited to, deep learning, random forest, regression analysis, naïve bayes, and K-nearest neighbours (KNN). It is vital to select the appropriate model as they all have their pros and cons and have their own use cases where they shine. When it comes to financing, four of them stand out: random forest, naïve bayes, decision tree, and KNN [12]. This report will provide a detailed description of random forest and its examples, introduce the reader to naïve bayes and KNN while comparing all these methods amongst each other, as well as how to optimise the code for random forest to achieve the best results.

2 Overview of Random Forest

Tin Kam Ho was the first to propose a random forest model in his work of 1995 [8]. However, the implementation of this idea had not been realised in its full glory until Leo Brieman published his article "Random Forest" in the "Machine Learning" journal in 2001 [2]. Random forest is an algorithm that combines the results of several decision trees into a single output. It found many use cases in banking, stock trading, medicine, and commerce by data scientists. Additionally, there are a number of significant benefits: • Reduced overfitting and improved accuracy compared to decision trees, • Great compatibility with both categorical and continuous values, • Automated missing values in a dataset, • Normalisation is not required. To determine whether a customer is trustworthy and can be lent to, random forest is a splendid pick as the problem falls into a categorical model with two classes: "Approved" and "Denied".

3 Background: Decision Trees

Any forest starts with a tree, and in the case of a random forest, it starts with a decision tree. A decision tree discovers patterns by learning simple decision rules derived from the training data. It utilises a hierarchical structure, using a root node, branches, and internal and leaf nodes. The deeper a decision tree grows the more sophisticated patterns are uncovered. However, this also tends to lead to overfitting, a case where a model starts blindly memorising patterns and does not truly learn. Therefore, when a new dataset is introduced to the algorithm that was not seen before, the chance of error increases exponentially. Nonetheless, a simple remedy for the issue would be utilisation of multiple such trees. The model used for determining whether a loan should be approved is a primitive version of random forest. The only parameters that change are the number of trees (n_estimators), and training and testing datasets. The model learns from the training dataset by growing N number of trees, and is then tested using the test dataset. However, there is more space for experimentation. Specifically, the documentation webpage for the random forest classifier has a long list of parameters that can be adjusted for various reasons[5]. Nonetheless, it is usually a balancing act between speed and accuracy. When one of them increases, the other one tends to decrease as shown in Figure 1. The decision lies with the client (bank), and they determine what they prefer.

This report will analyse all possible tweaks in the model that could help improve one of these parameters. An experiment was done where all parameters from the documentation were altered, and the results were recorded in Figure 2.

4 Description of Parameters

Some of the parameters in the experiment are quite intuitive due to their name and corresponding change. For example, max_leaf_nodes defines the number of leaves a tree can have. This parameter prevents overfitting. Similarly, max_depth sets the depth of a tree. Nonetheless, some variables might require more explanation. For instance,

```
start_time = time.time()
random_forest = RandomForestClassifier(n_estimators=100, max_depth=10)
random_forest.fit(X_train, y_train)
pred_rf = random_forest.predict(X_test)
acc_rf = accuracy_score(y_test , pred_rf)*100
print(acc_rf)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
83.78378378378379
--- 0.18943357467651367 seconds ---
```

```
start_time = time.time()
random_forest = RandomForestClassifier(n_estimators=100, max_depth=5)
random_forest.fit(X_train, y_train)
pred_rf = random_forest.predict(X_test)
acc_rf = accuracy_score(y_test , pred_rf)*100
print(acc_rf)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
83.24324324324324
--- 0.17961621284484863 seconds ---
```

Figure 1. Comparison of time when max_depth = 10 (up) and 5 (down).

n_jobs sets the number of processes (such as fit, and predict) to run in parallel. Warm_start, if set to True, reuses the solution of the previous call to fit. Bootstrap, by default, is True, samples random subsets from a dataset. Finally, min_impurity_decrease splits a node if the reduction of the impurity is greater than the value of this variable.

Tweak/Test	1	2	3	4	5	6	7	8	9	10	Average
max_depth=5	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	**83.2432**
max_leaf_nodes=5	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	**83.2432**
max_leaf_nodes=10	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	**83.2432**
min_impurity_decrease=0.01	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	**83.2432**
min_impurity_decrease=0.001	82.7027	83.2432	83.7837	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	83.2432	**83.2432**
max_depth=10	83.2432	82.7027	83.2432	82.1621	82.1621	82.7027	83.2432	82.1621	82.7027	82.7027	**82.7027**
n_jobs=10	80.5405	81.081	80.5405	81.6216	79.4594	80.5405	82.1621	80	80.5405	80.5405	**80.7027**
n_jobs=1	81.081	80	80.5405	80.5405	80.5405	80.5405	80.5405	80.5405	80.5405	81.081	**80.5946**
warm_start=True	80.5405	81.081	80	80.5405	81.081	80.5405	79.4594	80.5405	80.5405	81.6216	**80.5946**
n_jobs=5	81.081	80	82.1621	81.081	80.5405	80.5405	80	80.5405	80	79.4594	**80.5405**
max_depth=100	80.0000	81.6216	81.0810	80.0000	80.5405	80.0000	80.0000	80.5405	80.0000	81.0810	**80.4865**
n_estimators = 1000	81.0810	80.5405	80.5405	80.5405	81.0810	79.4594	80.0000	80.5405	80.5405	80.5405	**80.4864**
n_estimators = 50	80.0000	80.0000	81.0810	81.0810	79.4594	81.0810	79.4594	81.6216	80.0000	80.0000	**80.3783**
Original	80.5405	80.5405	80.5405	81.0810	78.9189	78.9189	80.5405	80.5405	80.0000	81.6216	**80.3243**
n_estimators = 150	80.0000	79.4594	80.0000	80.0000	80.5405	81.0810	80.5405	80.5405	80.5405	80.0000	**80.2702**
n_jobs=-1	80.5405	80	80.5405	79.4594	80	79.4594	81.081	81.081	79.4595	81.0811	**80.2702**
bootstrap=False	78.3783	77.2972	79.4594	77.2972	78.9189	79.4594	79.4594	80	79.4594	77.2972	**78.7026**
n_estimators = 10	78.9189	76.7567	74.5945	78.3783	77.8378	75.1351	81.0810	75.6756	77.8378	79.4594	**77.5675**
max_depth=1	83.2432	78.9189	70.2702	76.7567	78.3783	78.9189	70.2702	70.2702	83.2432	78.3783	**76.8648**
max_leaf_nodes=2	70.2702	72.9729	80.0000	70.2702	70.2702	70.8108	80.5405	70.2702	70.2702	83.2432	**73.8918**
min_impurity_decrease=0.1	70.2702	70.2702	70.2702	70.2702	70.2702	70.2702	70.2702	70.2702	70.2702	70.2702	**70.2702**

Figure 2. All possible modifications to the model.

5 Improving Model

To ensure the fairness of the experiment, after each alteration, the code was run 10 times, and the average results were compared. The test runs of the original model demonstrate this effect. The results vary from 78.91% to 81.62%. The disparity is even greater when changing the parameter max_depth=1 the lowest outcome is 70.27%, but the highest is 83.24%. When comparing the highest result of this alteration and the accuracy was 83.24%, one could think that setting max_depth=1 would be the right move. However, on average, this change would worsen the predicting ability of the model. As can be seen in Figure 2, the best accuracy was achieved by setting max_depth equal to 5, max_leaf_nodes – to 5 or 10, and min_impurity_decrease – to 0.01. However, even after combining a couple/several of these alterations, the accuracy was capped at 83.24%. Despite all possible combinations of these changes, the accuracy remained at 83.24%.

6 Ensemble Methods

Ensemble Methods (from French “ensemble” – “together” [1]) is a technique that combines several models to achieve the most optimal and accurate solution [4]. In fact, a random forest is considered to be a prime example of ensemble methods as it combines an average answer from a number of decision trees. However, the random forest itself can also be combined with several other models for this purpose. This technique is explored in-depth by Gayathri D. R. and Sumanjani P[7]. In short, they combined the results of naïve bayes, random forest, and K-nearest neighbours, and achieved an accuracy of 0.98%.

7 Bias

AI can significantly help improve the quality of the service – the customers can request a loan online, and because AI does not demand any salary or rest, and it can answer inquiries about credit 24/7. It can also reduce bias. Multiple studies proved that appearance matters to humans, and bank tellers are no exception [13]. While most of them are aware of this effect, some biases might be on an unconscious level, and go unnoticed. Nonetheless, AI could also be affected by this. As AI tries to discover patterns and correlations, it could lead to discrimination against some groups. For example, the trained random forest model shows that applicants with a better credit history have higher chances of being accepted. It also favours employees over self-employed. These biases can be affected by various factors, but usually - by training datasets, where, for instance, all female applicants are rejected (which is not the case in this dataset). Therefore, the model could conclude that all females have to be denied any credit. This is an extreme case, and real-life situations are more subtle. However, they still have huge impacts on such groups. Thus, all AIs have to be carefully trained and analysed to prevent any discrimination and unfairness.

8 Conclusion and future work

Random Forest model based on Decision Trees. However, it overcomes overfitting. Its primitive version can yield high results at around 80%. It could be further increased by 3% by tweaking additional parameters. Nonetheless, it is best to be combined with other algorithms utilising ensemble methods to potentially increase the accuracy to 98%. This could be a ground for future work.

REFERENCES

[1] Henri Mitterand Albert Dauzat, Jean Dubois, *Nouveau dictionnaire étymologique*, Librairie Larousse, 1964.
[2] Leo Breiman, ‘Random forests’, *Machine learning*, **45**, 5–32, (2001).
[3] Google Cloud. What is artificial intelligence (ai)?
[4] David Cournapeau. Ensembles: Gradient boosting, random forests, bagging, voting, stacking.
[5] David Cournapeau. sklearn.ensemble.randomforestclassifier.
[6] Joe Decosmo, ‘Artificial intelligence across the lending life cycle’, *Forbes*, (2023).
[7] R Gayathri Devi and P Sumanjani, ‘Improved classification techniques by combining knn and random forest with naive bayesian classifier’, in *2015 IEEE International Conference on Engineering and Technology (ICETECH)*, pp. 1–4. IEEE, (2015).
[8] Tin Kam Ho, ‘Random decision forests’, in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pp. 278–282 vol.1, (1995).
[9] IBM. What is artificial intelligence (ai)?
[10] VARTEQ Inc. Ai in credit scoring: Enhancing loan approvals and financial inclusion through advance algorithms.
[11] Oxford University Press, *Artificial Intelligence*, Oxford UP, 2023.
[12] Vishwas K N. Viswanatha V., Ramachandra A.C. and Adithya G, ‘Prediction of loan approval in banks using machine learning approach’, *International Journal of Engineering and Management Research*, **13**, 5–32, (2023).

[13] Leslie A Zebrowitz and Joann M Montepare, 'Social psychological face perception: Why appearance matters', *Social and personality psychology compass*, **2**(3), 1497–1517, (2008).