Campus Querétaro

Random Forest and Neuronal Network Using Framework

Gamaliel Marines Olvera
A01708746

Advanced AI

September 3, 2024

Introduction

My name is Gamaliel Marines, and in this academic paper, I explore a machine learning model called Random Forest. For this study, I choose to work with a supervised learning model designed for prediction. I use a specific framework to develop a functional and accurate model.

Context of the Problem

The challenge I aim to address with the implementation of this model is the inconsistency in credit approval processes at a bank in Europe. This research provides a better understanding of clients and lays the foundation for optimizing human resource allocation to improve customer service, as well as enhancing the efficiency of resource distribution, such as advertising and other customer engagement strategies.

Background on the domain and problem being addressed

The problem revolves around credit approval at a bank in Europe, specifically in Portugal. The dataset used for this study was collected from UC Irvine.

Significance of the problem in a real-world context

This research can help banks and loan service enterprises worldwide understand the importance of optimizing resource distribution to maximize operational efficiency throughout the financial year.

Ethics of the problem

It is crucial to recognize the potential ethical concerns associated with the misuse of this model. If misapplied, it could lead to the discrimination of specific age groups or other demographic groups, resulting in negative social consequences.

Objectives and Research Questions

Clear definition of the objectives of the study

The main objective of this project is to develop accurate predictions through the training of mathematical models and to explain their functionality and use. Additionally, the project aims to highlight the relevance of these models for enterprises and businesses.

Key questions the paper aims to answer

This paper aims to answer several key questions, such as:

- How do hyperparameters affect the performance of machine learning models?
- How does the performance and efficiency of a Random Forest compare to that of a Neural Network?
- When is it more appropriate to use a Random Forest versus a Neural Network?

Overview of Machine Learning Approaches

Brief overview of Random Forests and Neuronal Network and their relevance to the problem

I employ two different machine learning techniques: Random Forests and Neural Networks. These methods are widely used for classification predictions due to their high accuracy.

I choose Random Forests because they build multiple decision trees during training and merge them to produce accurate and stable predictions. They are effective and can easily mitigate overfitting, which is a common issue in machine learning models. The Random Forest method is well-suited for this problem because it can handle complex relationships between features and its built-in feature selection helps identify the most important variables influencing the outcome.

I also choose Neural Networks because of their effectiveness for handling large datasets with complex and non-linear relationships. They are particularly well-suited for this problem because they can model intricate patterns that simpler models might miss, potentially providing more accurate predictions.

Both methods offer distinct advantages: Random Forests provide interpretability and are less prone to overfitting, while Neural Networks can capture complex, non-linear relationships in the data. By comparing these two methods, I aim to determine which approach offers higher accuracy and better generalization for the specific classification problem at hand.

Dataset Preparation

Choosing and preparing the dataset is a crucial step in developing a reliable machine learning model. Proper data preparation ensures that the model is trained on high-quality data, which directly impacts its performance and accuracy. The dataset preparation process involves several stages: data collection and description, data cleaning and preprocessing, and dataset partitioning.

Data Collection and Description

Sources of the dataset and its main features

The data used in this study comes from UC Irvine. The dataset includes a variety of features that are essential for the prediction task. For instance, in the case of credit approval, the features include client age, job, marital, education, income, housing, credit history. Description of the data structure, feature types, and any initial insights

The dataset is structured in a tabular format, with rows representing individual instances and columns representing different features. The feature types can vary: some are numerical, while others are categorical. Initial exploration of the data may reveal patterns, such as correlations between certain features and credit approval outcomes, or imbalances in the data that need to be addressed.

Data Cleaning and Preprocessing

Methods for handling missing values, outliers, and incorrect data

Data cleaning involves identifying and handling missing values, outliers, and incorrect data entries. Missing values can be managed by either imputing them with statistical measures like mean, median, or mode or by removing the records entirely if they are insignificant. Outliers are detected using statistical methods, such as Z-scores or in this case interquartile ranges (IQR), and can be handled by transformation, capping, or removal, depending on their impact on the model. For this study I use IQR to identify outliers and winsorization to handle those outliers.

Feature Engineering for Model Enhancement

Feature engineering involves creating new features or transforming existing ones to improve the predictive power of a model. Transformations, such as normalizing or standardizing numerical features, are essential to ensure that the model treats all features equally, particularly when they exist on different scales. For this study, I employed two key techniques: scaling and One Hot Encoding.

- **Scaling:** This technique is applied to numerical features to bring them to a common scale. By standardizing these features I ensure that the model does not disproportionately weigh features with larger numerical ranges. This is crucial because the scale of the features can significantly impact model performance and convergence speed.
- **One Hot Encoding:** This technique is used to transform categorical features into a format suitable for machine learning algorithms. Each category is converted into a new binary feature, where a value of 1 indicates the presence of that category and 0 its absence.

Explanation of the decision to remove or keep specific data points or features

Decisions to remove or retain specific data points or features are made based on their relevance to the problem and their influence on model performance. Features that are irrelevant, redundant, or highly correlated with others may be removed to simplify the model and reduce the risk of overfitting. Similarly, data points that are identified as noise or have a high probability of being errors may be excluded to improve the model's accuracy. For this study I removed the feature: contact.

Dataset Partitioning

Once the data is cleaned and preprocessed, it is partitioned into different sets to evaluate the model's performance effectively. I split the dataset into three subsets: training, validation, and test sets. The training set is used to train the model, the validation set is used for tuning hyperparameters and preventing overfitting, and the test set is used to evaluate the model's generalization performance on unseen data. Proper partitioning ensures that the model is robust and performs well across different data samples.

Rationale behind the chosen split ratios

The chosen split ratios (70% for the training set, 20% for the validation set, and 10% for the test set) are designed to balance the needs for robust model training, effective hyperparameter tuning, and reliable evaluation.

- **70% Training Set:** Allocating 70% of the data for training ensures that the model has enough examples to learn effectively from a diverse range of samples, reducing the risk of underfitting.
- **20% Validation Set:** Using 20% of the data for validation allows sufficient opportunities to tune the model's hyperparameters and monitor its performance for signs of overfitting, ensuring that it generalizes well to new data.

- **10% Test Set:** A smaller test set of 10% is sufficient to evaluate the model's final performance, given that it is only used once and should represent the entire data distribution fairly well.

Methodology

Overview of Random Forest Algorithm

Description of the mathematical model and key concepts: decision trees, bagging, and feature importance

Decision Trees: A decision tree is a flowchart-like structure in which each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label (e.g., credit approved or denied). The model makes decisions by traversing the tree from the root to a leaf, based on feature values of the input data.

Bagging: Bagging is a technique used in Random Forests to reduce variance and improve model accuracy. It involves generating multiple subsets of the original data by random sampling with replacement. Each decision tree in the forest is trained on a different subset of data, and the final prediction is obtained by averaging (for regression) or voting (for classification) across all trees. This approach ensures that each tree has different training data, leading to diverse models that together provide a more robust prediction.

Hyperparameter Tuning

Description of the hyperparameters

Random Forest has several hyperparameters that need to be optimized to achieve the best model performance:

- **Number of Trees (n_estimators)**: The number of decision trees in the forest. A higher number generally leads to better performance but also increases computational cost.
- **Maximum Depth of Trees (max_depth)**: The maximum number of levels allowed in each decision tree. Controlling the depth helps prevent overfitting by restricting the complexity of the trees.
- **Minimum Samples per Leaf (min_samples_leaf)**: The minimum number of samples required to be at a leaf node. This parameter prevents overfitting by ensuring that the tree does not become too specialized on small subsets of the data.
- **Maximum Number of Features (max_features)**: The maximum number of features considered for splitting at each node. This controls the diversity of trees in the forest and can improve generalization.

Methods for searching the best hyperparameters

Finding the optimal hyperparameters is critical for improving model performance. Several methods are used to search for the best hyperparameters:

- **Grid Search**: A method that exhaustively searches through a manually specified subset of the hyperparameter space. It evaluates all possible combinations of hyperparameters and selects the one with the best performance based on a chosen evaluation metric.
- **Random Search**: Instead of searching all combinations, Random Search randomly selects a combination of hyperparameters from a specified distribution. This approach is often more efficient than Grid Search, as it explores the space more broadly.
- **Bayesian Optimization**: A more advanced method that builds a probabilistic model of the objective function and uses it to select the most promising hyperparameters to evaluate next. It is particularly useful for optimizing hyperparameters when the search space is large and evaluations are costly.

For the purpose of this academic paper, which aims to deepen the understanding of the methodology, I opted not to use these automated hyperparameter optimization techniques. Instead, I manually searched for and tuned the hyperparameters, documenting the values and results obtained. This approach allowed for a more hands-on exploration of the hyperparameter space and provided a deeper insight into the impact of each parameter on the model's performance.

Explanation of the metrics used to evaluate model performance

To assess the performance of the Random Forest model, several evaluation metrics are used:

- **Accuracy**: The proportion of correctly predicted instances out of the total instances. While accuracy provides a general measure of performance, it may not be sufficient when dealing with imbalanced datasets.
- **Precision**: The proportion of true positive predictions among all positive predictions. It measures how many of the predicted positive cases are actually positive. Precision is particularly important when the cost of false positives is high (e.g., wrongly approving a bad credit).
- **Recall (Sensitivity)**: The proportion of true positive predictions among all actual positive cases. It measures the ability of the model to identify all positive cases. Recall is crucial when the cost of false negatives is high (e.g., missing out on a good credit approval).
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives. It is useful when there is an uneven class distribution.

Methodology for Neural Network

Overview of Neural Networks

Neural networks are computational models inspired by the human brain, designed to recognize patterns in data. They consist of layers of interconnected nodes (neurons) that learn to map input features to output targets by adjusting weights through training. Neural networks are particularly well-suited for handling complex and nonlinear relationships in data, which makes them ideal for tasks such as credit approval.

In this case, I use a **fully connected neural network** (also known as a multilayer perceptron, MLP) to predict credit approval. The network has an input layer, two hidden layers, and an output layer, with a Dropout mechanism added to prevent overfitting. Dropout randomly sets a fraction of input units to zero during training, which forces the model to learn more robust features that generalize better.

Description of the Neural Network Model and Key Concepts

- **Dense Layers**: These are fully connected layers where each neuron is connected to all the neurons in the previous and subsequent layers. I use two hidden layers with 64 and 32 neurons, respectively. Each hidden layer uses the sigmoid activation function to introduce non-linearity into the model, allowing it to learn complex patterns in the data.
- **Dropout**: A regularization technique to prevent overfitting by randomly "dropping out" a fraction of input units during the training process. This forces the model to not rely on any single input, making it more robust. In this model, I apply a dropout rate of 50% after each hidden layer.
- **Output Layer**: The output layer has 2 neurons corresponding to the two classes (approved or denied) in our binary classification problem. The `softmax` activation function is used to output probabilities for each class.

Architecture of the Neural Network

The network consists of:

1. An **input layer** that takes the feature set (`X_train`) with the input dimension equal to the number of features.
2. Two **hidden layers**:
    - **First hidden layer**: 64 neurons with the sigmoid activation function.
    - **Second hidden layer**: 32 neurons with the sigmoid activation function.
    - **Dropout layers** after each hidden layer with a dropout rate of 50%.
3. An **output layer** with 2 neurons for the binary classification task, using the softmax activation function.

Hyperparameters and Optimization

- **Learning Rate**: Controls the step size during optimization. I use a learning rate of 0.001 with the Adam optimizer, which combines the advantages of two popular optimizers: AdaGrad and RMSProp.
- **Batch Size**: The number of samples processed before the model's internal parameters are updated. A batch size of 32 is used for efficient computation.
- **Number of Epochs**: The number of complete passes through the training dataset. I use 50 epochs to ensure sufficient learning while preventing overfitting.

Methods for Improving Model Performance

- **Dropout Regularization**: Used to mitigate overfitting by randomly disabling a fraction of neurons during training.
- **Early Stopping (not implemented)**: Could be employed to stop training when the validation performance stops improving, thus preventing overfitting.
- **Batch Normalization**: Can help accelerate training and improve stability by normalizing the inputs for each layer.

Evaluation Metrics for Model Selection

To evaluate the performance of the neural network model, I use the following metrics:

- **Accuracy**: Measures the proportion of correctly predicted instances among the total instances. It provides a general sense of how well the model is performing.
- **Precision**: Calculates the proportion of true positives among all predicted positives, useful when the cost of false positives is high.
- **Recall**: Calculates the proportion of true positives among all actual positives, important when the cost of false negatives is high.
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of both metrics.
- **Confusion Matrix**: Provides a visual representation of the model's performance by showing the count of true positive, true negative, false positive, and false negative predictions.
- **Loss Plot**: Visualizes the evolution of the training and validation loss over epochs, helping in diagnosing overfitting or underfitting.

Implementation

Code Design and Architecture

Description of the code structure and components.

The code is structured into several key components that cover the entire process of data preprocessing, model building, training, evaluation, and comparison between two machine learning algorithms: Random Forest and a Neural Network. The components of the code are organized as follows:

1. **Data Loading and Preprocessing**:
   - The dataset is loaded using `pandas` and cleaned by dropping irrelevant columns and converting categorical variables to a numerical format through one-hot encoding.
   - Missing or incorrect data values are handled by converting them into numerical types.
2. **Dataset Partitioning**:
   - The cleaned dataset is divided into training, validation, and test sets using the `train_test_split` function from `sklearn`.
3. **Model Building**:
   - A Random Forest model is built using `RandomForestClassifier` from `sklearn.ensemble`.
   - A Neural Network model is created using the Keras Sequential API, with several layers added to handle complex patterns in the data.
4. **Model Training and Evaluation**:
   - Both models are trained on the training set, and their performances are evaluated on the validation and test sets using metrics like accuracy, precision, recall, and F1-score.
   - Confusion matrices and classification reports are generated for each model to understand their performance on the test set.
5. **Results Comparison**:
   - The performance metrics of both models (Random Forest and Neural Network) are compared to identify which one offers better generalization and accuracy for the classification problem.

Libraries and tools used for implementation

Python Libraries:

- `numpy` and `pandas` for data handling and manipulation.
- `matplotlib` and `seaborn` for data visualization and plotting graphs.
- `scipy` for hierarchical clustering and data analysis.
- `scikit-learn` for machine learning algorithms, model evaluation metrics, and hyperparameter tuning.
- `tensorflow.keras` for building and training the Neural Network model.

Workflow and Execution

Data Loading and Exploration

- The dataset (`bank.csv`) is loaded using `pandas` and basic information about the dataset is displayed (`df.info()` and `df.head()`).

Data Preprocessing

- Irrelevant columns are removed.
- Categorical variables are converted to numerical format using one-hot encoding.
- Boolean columns are converted to integers for compatibility with machine learning models.

Dataset Partitioning

- The dataset is split into training (70%), validation (20%), and test sets (10%). This step ensures that the model is trained on one portion of the data and validated on another before testing on the final, unseen portion.

Model Building

- **Random Forest Model**: A `RandomForestClassifier` is initialized with specified hyperparameters (`n_estimators`, `max_depth`, `min_samples_split`, etc.) and trained on the training data.
- **Neural Network Model**: A `Sequential` model is built using Keras, comprising an input layer, one hidden layer, and an output layer suitable for binary classification. The model is compiled with an optimizer (Adam), loss function (categorical cross-entropy), and evaluation metric (accuracy).

Model Training

- The Random Forest model is trained using the `fit` method on the training data.
- The Neural Network is trained using the `fit` method on the training data, with validation data provided to monitor performance during training.

Model Evaluation

- The accuracy and classification report are computed for both the Random Forest and Neural Network models on training, validation, and test datasets.
- Confusion matrices are plotted to visualize the performance of each model in terms of correctly and incorrectly classified samples.

Results Comparison

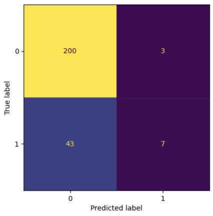
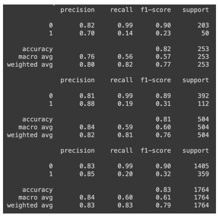
- A direct comparison between the Random Forest and Neural Network models is made based on their accuracy and other metrics on the test dataset. The model with higher accuracy and better generalization performance is highlighted.

Results and Discussion

Random Forest Result Table

<table>
<tr><th>Sampling</th><th>Hyperparameters</th><th>Accuracy</th><th>Confussion Matrix (CM)</th><th>Interpretation</th></tr>
<tr>
<td>training: 1764
validation: 504
test: 253</td>
<td>trees: 50
max depth: 10
min sample split: 2
max_leaf_nodes: 10
n_jobs: 1
random state: 42</td>
<td>Training Accuracy: 0.8299

Validation Accuracy: 0.8135

Test Accuracy: 0.8182</td>
<td>
</td>
<td>
<pre>
              precision   recall f1-score   support

           0   0.82   0.99   0.90   203
           1   0.70   0.14   0.23   50

    accuracy   0.82   253
   macro avg   0.76   0.56   0.57   253
weighted avg   0.80   0.82   0.77   253
              precision   recall f1-score   support

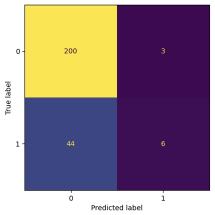
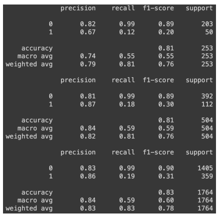
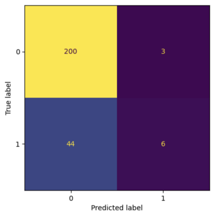
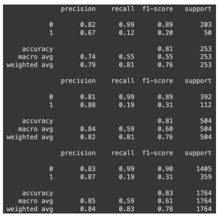
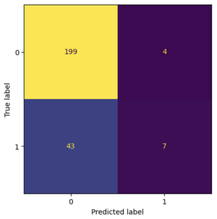
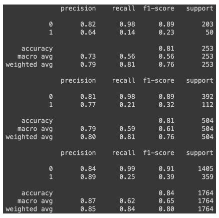
           0   0.81   0.99   0.89   392
           1   0.88   0.19   0.31   112

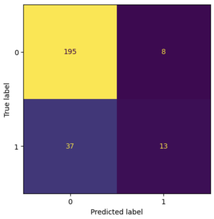
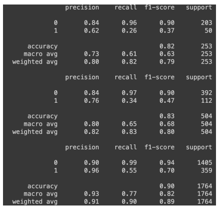
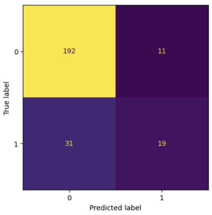
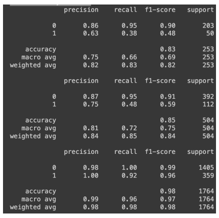
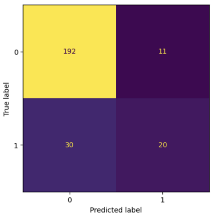
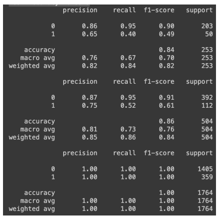
    accuracy   0.81   504
   macro avg   0.84   0.59   0.60   504
weighted avg   0.82   0.81   0.76   504
              precision   recall f1-score   support

           0   0.83   0.99   0.90   1405
           1   0.85   0.20   0.32   359

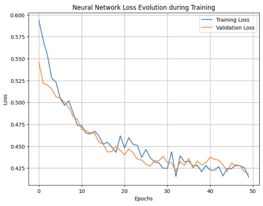
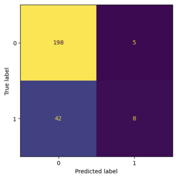
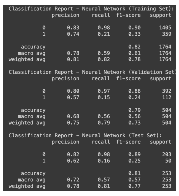
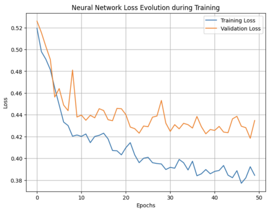
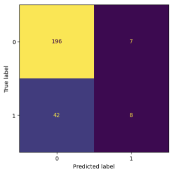
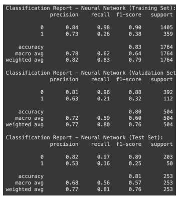
    accuracy   0.83   1764
   macro avg   0.84   0.60   0.61   1764
weighted avg   0.83   0.83   0.79   1764
</pre>
using few trees, depth and leaves we reach over 80% of accuracy.

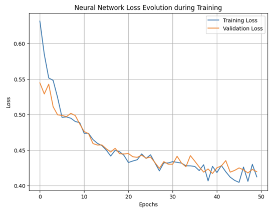
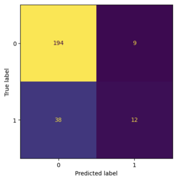
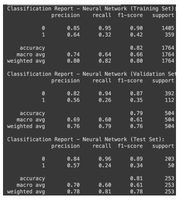
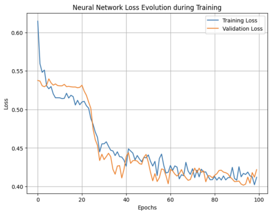
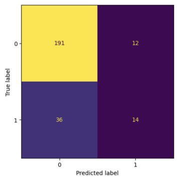
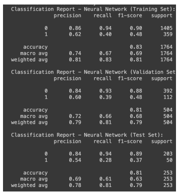
However, these results are not as honest as they appear. There is great bias towards negative results (shown in the CM).</td>
</tr>
</table>

training: 1764 validation: 504 test: 253	trees: 1000 max depth: 10 min sample split: 2 max_leaf_nodes: 10 n_jobs: 1 random state: 42	Training Accuracy: 0.8282 Validation Accuracy: 0.8115 Test Accuracy: 0.8142		 precision recall f1-score support 0 0.82 0.99 0.89 203 1 0.67 0.12 0.20 50 accuracy 0.81 253 macro avg 0.74 0.55 0.55 253 weighted avg 0.79 0.81 0.76 253 precision recall f1-score support 0 0.81 0.99 0.89 392 1 0.87 0.18 0.30 112 accuracy 0.81 504 macro avg 0.84 0.59 0.59 504 weighted avg 0.82 0.81 0.76 504 precision recall f1-score support 0 0.83 0.99 0.90 1405 1 0.86 0.19 0.31 359 accuracy 0.83 1764 macro avg 0.84 0.59 0.60 1764 weighted avg 0.83 0.83 0.78 1764 By increasing the trees, I observe a better accuracy level. The negative bias continues.
training: 1764 validation: 504 test: 253	trees: 500 max depth: 10 min sample split: 2 max_leaf_nodes: 10 n_jobs: 1 random state: 42	Training Accuracy: 0.8294 Validation Accuracy: 0.8135 Test Accuracy: 0.8142		 precision recall f1-score support 0 0.82 0.99 0.89 203 1 0.67 0.12 0.20 50 accuracy 0.81 253 macro avg 0.74 0.55 0.55 253 weighted avg 0.79 0.81 0.76 253 precision recall f1-score support 0 0.81 0.99 0.89 392 1 0.88 0.19 0.31 112 accuracy 0.81 504 macro avg 0.84 0.59 0.60 504 weighted avg 0.82 0.81 0.76 504 precision recall f1-score support 0 0.83 0.99 0.90 1405 1 0.87 0.19 0.31 359 accuracy 0.83 1764 macro avg 0.85 0.59 0.61 1764 weighted avg 0.84 0.83 0.78 1764 An acceptable accuracy level is maintained although the number of trees is cut in half. The negative bias continues.
training: 1764 validation: 504 test: 253	trees: 200 max depth: 20 min sample split: 2 max_leaf_nodes: 20 n_jobs: 1 random state: 42	Training Accuracy: 0.8418 Validation Accuracy: 0.8095 Test Accuracy: 0.8142		 precision recall f1-score support 0 0.82 0.98 0.89 203 1 0.64 0.14 0.23 50 accuracy 0.81 253 macro avg 0.73 0.56 0.56 253 weighted avg 0.79 0.81 0.76 253 precision recall f1-score support 0 0.81 0.98 0.89 392 1 0.77 0.21 0.32 112 accuracy 0.81 504 macro avg 0.79 0.59 0.61 504 weighted avg 0.80 0.81 0.76 504 precision recall f1-score support 0 0.84 0.99 0.91 1405 1 0.89 0.25 0.39 359 accuracy 0.84 1764 macro avg 0.87 0.62 0.65 1764 weighted avg 0.85 0.84 0.80 1764 A better level of accuracy is reached although the number of trees is once again reduced, but the depth and number of leaves is increased. The negative bias continues.

training: 1764 validation: 504 test: 253	trees: 500 max depth: 50 min sample split: 2 max_leaf_nodes: 50 n_jobs: 1 random state: 42	Training Accuracy: 0.9048 Validation Accuracy: 0.8294 Test Accuracy: 0.8221		 precision recall f1-score support 0 0.84 0.96 0.90 203 1 0.62 0.26 0.37 50 accuracy 0.82 253 macro avg 0.73 0.61 0.63 253 weighted avg 0.80 0.82 0.79 253 precision recall f1-score support 0 0.84 0.97 0.90 392 1 0.76 0.34 0.47 112 accuracy 0.83 504 macro avg 0.80 0.65 0.68 504 weighted avg 0.82 0.83 0.80 504 precision recall f1-score support 0 0.90 0.99 0.94 1405 1 0.96 0.55 0.70 359 accuracy 0.90 1764 macro avg 0.93 0.77 0.82 1764 weighted avg 0.91 0.90 0.89 1764 A significant increase in the accuracy level is obtained when the number of trees, depth and leaves is increased. The negative bias is less, however it remains.
training: 1764 validation: 504 test: 253	trees: 1000 max depth: 200 min sample split: 2 max_leaf_nodes: 200 n_jobs: 1 random state: 42	Training Accuracy: 0.9841 Validation Accuracy: 0.8492 Test Accuracy: 0.8340		 precision recall f1-score support 0 0.86 0.95 0.90 203 1 0.63 0.38 0.48 50 accuracy 0.83 253 macro avg 0.75 0.66 0.69 253 weighted avg 0.82 0.83 0.82 253 precision recall f1-score support 0 0.87 0.95 0.91 392 1 0.75 0.48 0.59 112 accuracy 0.85 504 macro avg 0.81 0.72 0.75 504 weighted avg 0.84 0.85 0.84 504 precision recall f1-score support 0 0.98 1.00 0.99 1405 1 1.00 0.92 0.96 359 accuracy 0.98 1764 macro avg 0.99 0.96 0.97 1764 weighted avg 0.98 0.98 0.98 1764 By doubling the number of trees and quadrupling the depth and leaves, the accuracy level increases significantly. The negative bias is less, however it remains.
training: 1764 validation: 504 test: 253	trees: 10,000 max depth: 500 min sample split: 2 max_leaf_nodes: 500 n_jobs: 1 random state: 42	Training Accuracy: 1.0000 Validation Accuracy: 0.8552 Test Accuracy: 0.8379		 precision recall f1-score support 0 0.86 0.95 0.90 203 1 0.65 0.40 0.49 50 accuracy 0.84 253 macro avg 0.76 0.67 0.70 253 weighted avg 0.82 0.84 0.82 253 precision recall f1-score support 0 0.87 0.95 0.91 392 1 0.75 0.52 0.61 112 accuracy 0.86 504 macro avg 0.81 0.73 0.76 504 weighted avg 0.85 0.86 0.84 504 precision recall f1-score support 0 1.00 1.00 1.00 1405 1 1.00 1.00 1.00 359 accuracy 1.00 1764 macro avg 1.00 1.00 1.00 1764 weighted avg 1.00 1.00 1.00 1764

Neuronal Network Result Table

Sampling	Hyperparameters	Accuracy	Loss Plot	Confussion Matrix (CM)	Interpretation
training: 1764 validation: 504 test: 253	input layer: 64 neurons activation function: sigmoid drop out: 50% hidden layer: 32 neurons activation function: sigmoid drop out: 50% output layer: 2 neurons activation function: softmax epochs: 50 batch size: 32	Neural Network Training Accuracy: 0.8254 Neural Network Validation Accuracy: 0.7877 Neural Network Test Accuracy: 0.8103			Classification Report – Neural Network (Training Set): precision recall f1-score support 0 0.83 0.98 0.90 1405 1 0.74 0.21 0.33 359 accuracy 0.82 1764 macro avg 0.78 0.59 0.61 1764 weighted avg 0.81 0.82 0.78 1764 Classification Report – Neural Network (Validation Set precision recall f1-score support 0 0.80 0.97 0.88 392 1 0.57 0.15 0.24 112 accuracy 0.79 504 macro avg 0.68 0.56 0.56 504 weighted avg 0.75 0.79 0.73 504 Classification Report – Neural Network (Test Set): precision recall f1-score support 0 0.82 0.98 0.89 203 1 0.62 0.16 0.25 50 accuracy 0.81 253 macro avg 0.72 0.57 0.57 253 weighted avg 0.78 0.81 0.77 253 Both, training and validation loss, decrease as the epochs advance showing that the model is learning. Both errors decrease and are similar meaning there is no overfitting. The accuracy level is acceptable, however there is a bias towards the negative predictions (shown in the CM).
training: 1764 validation: 504 test: 253	input layer: 64 neurons activation function: sigmoid drop out: 20% hidden layer: 64 neurons activation function: sigmoid drop out: 20% output layer:	Neural Network Training Accuracy: 0.8294 Neural Network Validation Accuracy: 0.7976 Neural Network Test Accuracy: 0.8063			Classification Report – Neural Network (Training Set): precision recall f1-score support 0 0.84 0.98 0.90 1405 1 0.73 0.26 0.38 359 accuracy 0.83 1764 macro avg 0.78 0.62 0.64 1764 weighted avg 0.82 0.83 0.79 1764 Classification Report – Neural Network (Validation Set precision recall f1-score support 0 0.81 0.96 0.88 392 1 0.63 0.21 0.32 112 accuracy 0.80 504 macro avg 0.72 0.59 0.60 504 weighted avg 0.77 0.80 0.76 504 Classification Report – Neural Network (Test Set): precision recall f1-score support 0 0.82 0.97 0.89 203 1 0.53 0.16 0.25 50 accuracy 0.81 253 macro avg 0.68 0.56 0.57 253 weighted avg 0.77 0.81 0.76 253 By decreasing the dropout, the model starts overfitting and stops learning as it should (shown in the Loss Plot).

	2 neurons activation function: softmax epochs: 50 batch size: 32				
training: 1764 validation: 504 test: 253	input layer: 64 neurons activation function: sigmoid drop out: 50% hidden layer: 64 neurons activation function: sigmoid drop out: 50% output layer: 2 neurons activation function: softmax epochs: 50 batch size: 32	Neural Network Training Accuracy: 0.8248 Neural Network Validation Accuracy: 0.7897 Neural Network Test Accuracy: 0.8142			 Classification Report – Neural Network (Training Set): precision recall f1-score support 0 0.85 0.95 0.90 1405 1 0.64 0.32 0.42 359 accuracy 0.82 1764 macro avg 0.74 0.64 0.66 1764 weighted avg 0.80 0.82 0.80 1764 Classification Report – Neural Network (Validation Set precision recall f1-score support 0 0.82 0.94 0.87 392 1 0.56 0.26 0.35 112 accuracy 0.79 504 macro avg 0.69 0.60 0.61 504 weighted avg 0.76 0.79 0.76 504 Classification Report – Neural Network (Test Set): precision recall f1-score support 0 0.84 0.96 0.89 203 1 0.57 0.24 0.34 50 accuracy 0.81 253 macro avg 0.70 0.60 0.61 253 weighted avg 0.78 0.81 0.78 253 By increasing the drop out the overfitting problem is solved. Increasing the neurons in the hidden layer a higher accuracy level is reached.
training: 1764 validation: 504 test: 253	input layer: 64 neurons activation function: sigmoid drop out: 50% 1st hidden layer: 64 neurons activation function: sigmoid	Neural Network Training Accuracy: 0.8271 Neural Network Validation Accuracy: 0.8075 Neural Network Test Accuracy: 0.8183			 Classification Report – Neural Network (Training Set): precision recall f1-score support 0 0.86 0.94 0.90 1405 1 0.62 0.40 0.48 359 accuracy 0.83 1764 macro avg 0.74 0.67 0.69 1764 weighted avg 0.81 0.83 0.81 1764 Classification Report – Neural Network (Validation Set precision recall f1-score support 0 0.84 0.93 0.88 392 1 0.60 0.39 0.48 112 accuracy 0.81 504 macro avg 0.72 0.66 0.68 504 weighted avg 0.79 0.81 0.79 504 Classification Report – Neural Network (Test Set): precision recall f1-score support 0 0.84 0.94 0.89 203 1 0.54 0.28 0.37 50 accuracy 0.81 253 macro avg 0.69 0.61 0.63 253 weighted avg 0.78 0.81 0.79 253 Too much resources for such a little increase in the accuracy level.

	drop out: 50% 2nd hidden layer: 64 neurons activation function: sigmoid drop out: 50% 3rd hidden layer: 64 neurons activation function: sigmoid drop out: 50% 4th hidden layer: 64 neurons activation function: sigmoid drop out: 50% output layer: 2 neurons activation function: softmax epochs: 100 batch size: 32				

Interpretation of Results

A common observation for both models is the great inclination for negative results. This observation is caused by the nature of the dataset, a great imbalance towards the negative results causes the models to favor negative predictions. A way I could deal with this is by engineering the dataset and balance the ratio of positive and negative results. However, since this is an academic paper whose objective is to further understand the models and their power I weighted the value of having better accuracy against deepening my understanding of the hyperparameters and how they react with biased datasets.

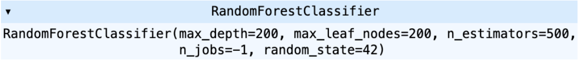
The Random Forest model and the Neural Network (NN) have both been evaluated on the task of credit approval prediction. The Random Forest model achieved higher accuracy on the test set compared to the Neural Network. This indicates that, in this case, the Random Forest model performs slightly better in predicting credit approvals.

In terms of precision, recall, and F1-score, the Random Forest model generally shows stronger performance for classifying both classes compared to the NN.

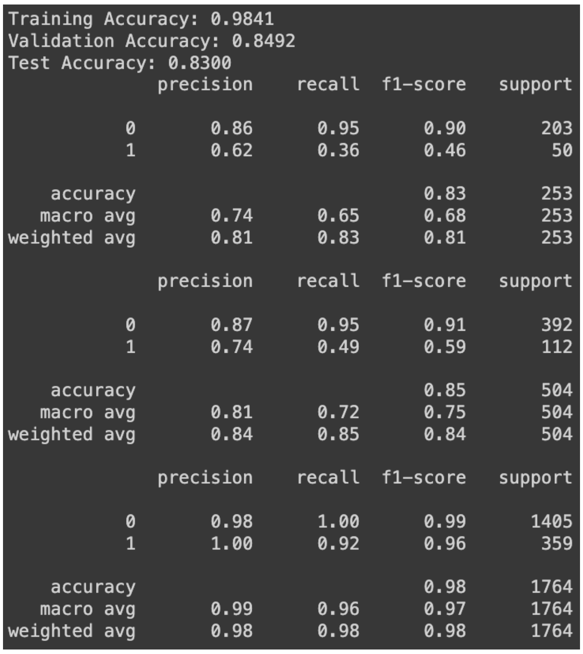
The Best Results Obtained

I choose these results because they have good accuracy levels while using "reasonable" hyperparameters and not abusing resources.

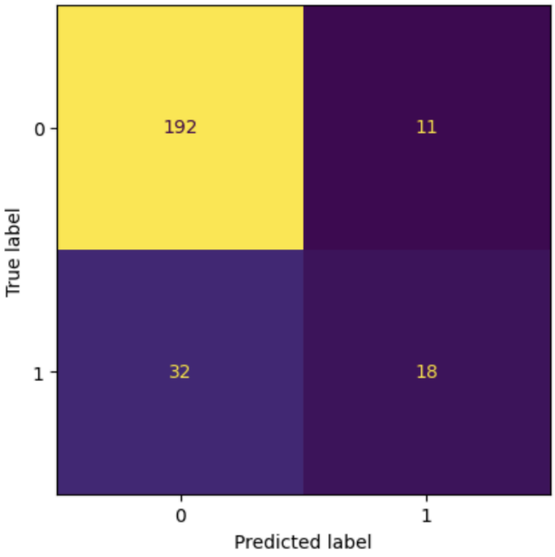
Random Forest Best Result



```
▾                        RandomForestClassifier
RandomForestClassifier(max_depth=200, max_leaf_nodes=200, n_estimators=500,
                       n_jobs=-1, random_state=42)
```

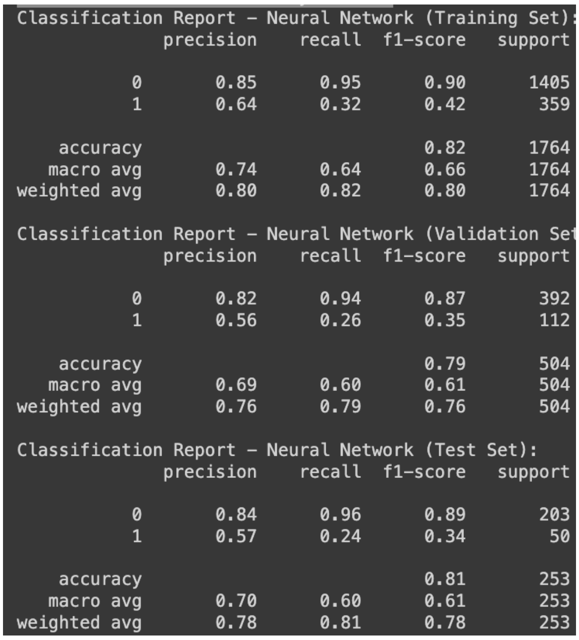


```
Training Accuracy: 0.9841
Validation Accuracy: 0.8492
Test Accuracy: 0.8300
              precision    recall  f1-score   support

           0       0.86      0.95      0.90       203
           1       0.62      0.36      0.46        50

    accuracy                           0.83       253
   macro avg       0.74      0.65      0.68       253
weighted avg       0.81      0.83      0.81       253

              precision    recall  f1-score   support

           0       0.87      0.95      0.91       392
           1       0.74      0.49      0.59       112

    accuracy                           0.85       504
   macro avg       0.81      0.72      0.75       504
weighted avg       0.84      0.85      0.84       504

              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1405
           1       1.00      0.92      0.96       359

    accuracy                           0.98      1764
   macro avg       0.99      0.96      0.97      1764
weighted avg       0.98      0.98      0.98      1764
```

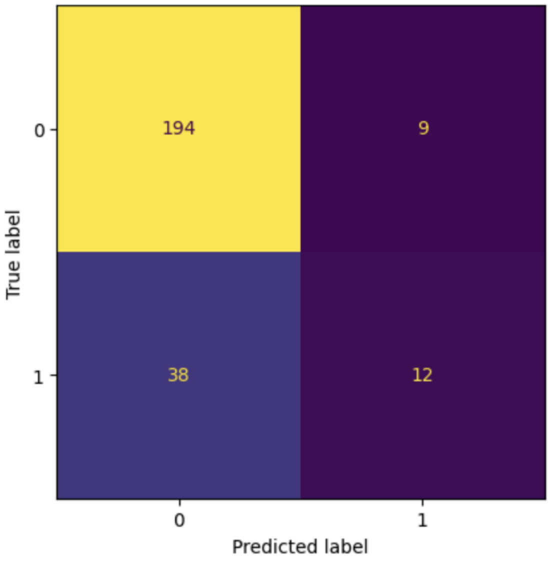


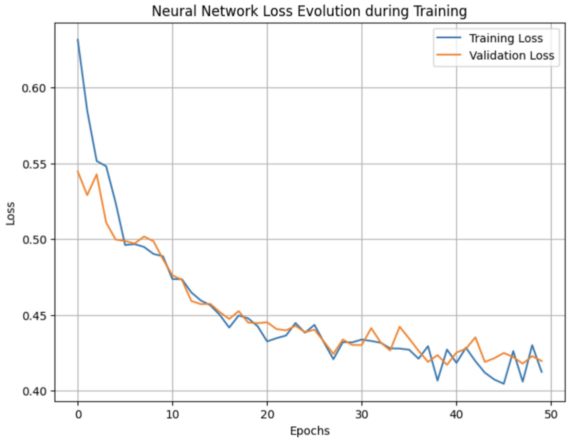
Neuronal Network Best Result

- input layer: 64 neurons, activation function: sigmoid, drop out: 50%
- hidden layer: 64 neurons, activation function: sigmoid, drop out: 50%
- output layer:2 neurons, activation function: softmax
- epochs: 50
- batch size: 32



```
Classification Report – Neural Network (Training Set):
              precision    recall  f1-score   support

           0       0.85      0.95      0.90      1405
           1       0.64      0.32      0.42       359

    accuracy                           0.82      1764
   macro avg       0.74      0.64      0.66      1764
weighted avg       0.80      0.82      0.80      1764

Classification Report – Neural Network (Validation Set
              precision    recall  f1-score   support

           0       0.82      0.94      0.87       392
           1       0.56      0.26      0.35       112

    accuracy                           0.79       504
   macro avg       0.69      0.60      0.61       504
weighted avg       0.76      0.79      0.76       504

Classification Report – Neural Network (Test Set):
              precision    recall  f1-score   support

           0       0.84      0.96      0.89       203
           1       0.57      0.24      0.34        50

    accuracy                           0.81       253
   macro avg       0.70      0.60      0.61       253
weighted avg       0.78      0.81      0.78       253
```





Insights Gained from the Model’s Performance and Feature Importance

The Random Forest’s higher performance can be attributed to its ensemble approach, which aggregates predictions from multiple decision trees, thereby improving robustness and accuracy. Its ability to handle high-dimensional data and various feature interactions makes it effective for this problem.

The Neural Network, despite its slightly lower accuracy, provides insights into the model's ability to learn complex patterns through its hidden layers.

Limitations and Potential Improvements

Random Forest Limitations

Overfitting: Although the Random Forest is less prone to overfitting compared to individual decision trees, it may still overfit, especially if the number of trees is not optimally chosen.

Neural Network Limitations

Performance Variability: The NN's performance is sensitive to hyperparameters, architecture choices, and training duration.

Class Imbalance: Both models show lower recall for class 1, indicating potential class imbalance issues. This suggests that the model struggles more with correctly identifying the minority class.

Suggestions for Further Research or Modifications to the Approach

Hyperparameter Tuning: Further tuning of hyperparameters, such as the number of trees in the Random Forest or the architecture of the Neural Network, could improve performance.

Handling Class Imbalance: Techniques such as resampling, class weighting, or using more advanced algorithms designed for imbalanced datasets could enhance model performance, especially for the minority class.

Feature Engineering: Exploring and incorporating additional features or more sophisticated feature selection methods could provide better insights and improve model accuracy.

Ensemble Methods: Combining the strengths of both models through ensemble methods, such as stacking or blending, might lead to improved overall performance.

Conclusion

Summary of Findings

The Random Forest model outperforms the Neural Network in terms of overall accuracy and class classification performance for credit approval prediction. The Random Forest's ensemble approach provides a robust model with better generalization, while the Neural Network, although less accurate, shows potential for learning complex patterns.

Future Research or Potential Extensions of the Study

Enhanced Hyperparameter Tuning: Employing advanced hyperparameter optimization techniques for both models.

Feature Expansion: Investigating additional or engineered features to improve model predictive power.

References

¿Qué es un bosque aleatorio? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/random-forest

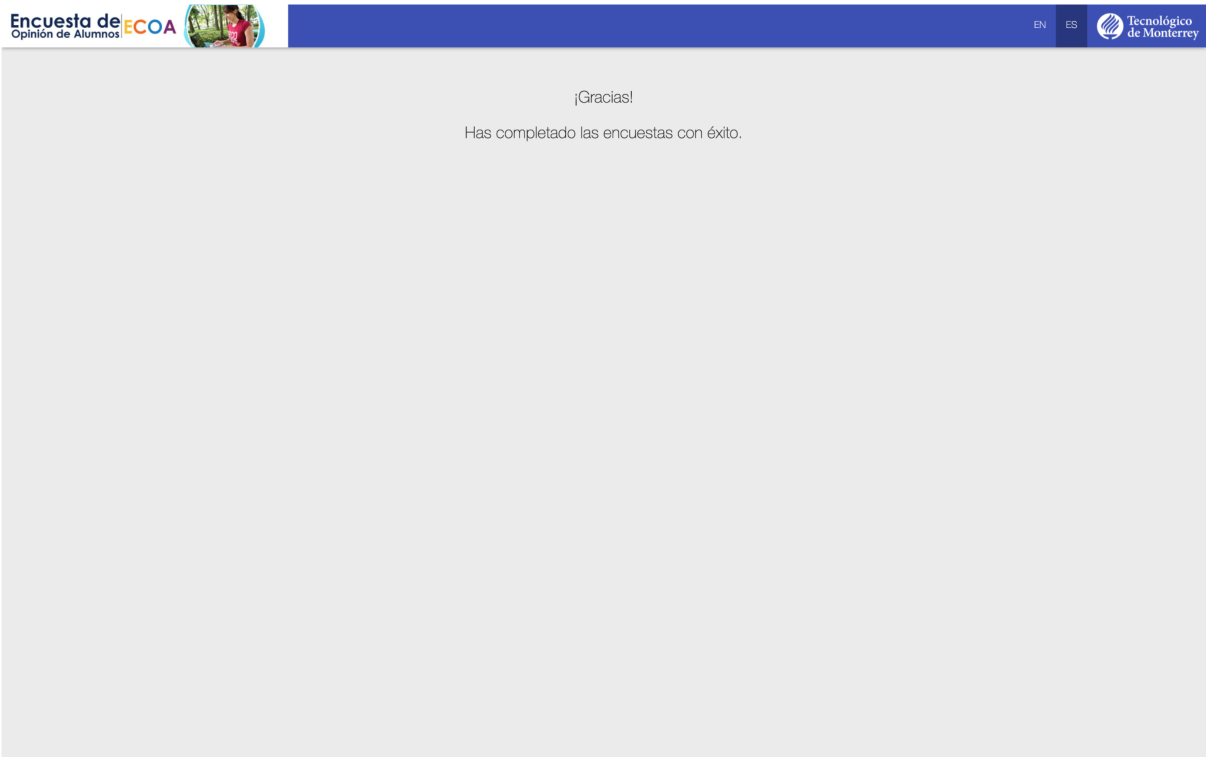
Daniel. (2023, 30 octubre). *Random Forest: Bosque aleatorio. Definición y funcionamiento*. Formación En Ciencia de Datos | DataScientest.com. https://datascientest.com/es/random-forest-bosque-aleatorio-definicion-y-funcionamiento

RandomForestClassifier. (s. f.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Index of Evidences

Evidencia	**Subcompetencia**	**Indicador**	**se observa indicador en la evidencia: si / no**
Portafolio Análisis	**SMA0102 Técnicas analíticas**	Utiliza al menos 2 ténicas de preprocesamiento de acuerdo al problema como escalamiento, detección de anomalias, imputación, etc...	1
		Explica claramente el uso de cada ténica de análisis utilizada y su relevancia en el set de datos.	1
	SMA0104 Análisis de información:	Evalúa el modelo con un conjunto de prueba y un conjunto de validación	1
		Detecta correctamente el grado de bias o sesgo: bajo medio alto	1
		Detecta correctamente el grado de varianza: bajo medio alto	1
		Explica el nivel de ajuste del modelo: underfitt fitt overfitt	1
		Utiliza técnicas de regularización para mejorar el desempeño del modelo	1
	SEG0403 Compromiso ético y ciudadano	Explica como la solución cumple leyes, normas y principios éticos, de la industria o el ambiente del reto.	1
		Explica en su repositorio cual es la normatividad correspondiente del reto o socio formador.	1
Evidencia	**Subcompetencia**	**Indicador**	**se observa indicador en la evidencia: si / no**
Portafolio Implementación	**SMA0101 Construcción de modelos**	Construye un modelo manualmente a partir de un set de datos, seleccionado las variables a utilizar.	1
		Explica correctamente cada una de las variables seleccionadas en el modelo y su utilidad en el modelo.	1
		Interpreta en detalle el modelo incluyendo los coeficientes y sus niveles de significancia estadística.	1
	SMA0401 Aprendizaje e IA	Implementa una técnica o algoritmo de aprendizaje máquina, sin uso de marco de trabajo o framework como regresiones, árboles, clusters, etc...	NA
		Usa un marco de trabajo o framework para implementa una técnica o algoritmo de aprendizaje máquina como: regresiones, árboles, clusters, etc...	1

ECOA



¡Gracias!

Has completado las encuestas con éxito.