Analysis of the Effectiveness of Using Machine Learning Algorithms to Make Hiring Decisions

Kirill Smelyakov [1], Yuliia Hurova [1] and Serhii Osiievskyi [2]

[1] *Kharkiv National University of Radio Electronics, 14 Nauky Ave., Kharkiv, 61166, Ukraine*
[2] *Kharkiv National University of Air Force, Ukraine*

Abstract

The focus of this paper is to analyze the effectiveness of modern machine learning algorithms in making hiring decisions for a team. By evaluating different criteria for each candidate, decision-makers can use these methods to make informed and objective hiring decisions. This study identifies various algorithms that can be used to solve this problem. Experiments are conducted to determine the most appropriate model for making hiring decisions using different machine learning algorithms. The experiments are conducted on a collected dataset that has been divided into training and test dataset. Conducting these experiments allows us to gain a deeper understanding of the data and draw more valid conclusions about the results of the study. The effectiveness of each algorithm is evaluated using several metrics, including processing time, accuracy, precision, recall, and F1 score. By analyzing and comparing these metrics, it is determined which of the algorithms is the most effective. The study culminates in a comparative analysis of metrics from the experiments, which provides valuable insight into the effectiveness of the different algorithms. Ultimately, this study helps decision-makers make better hiring decisions and lays the foundation for future research in this area.

Keywords

Recruitment Candidates, Hiring Candidates, Decision Rules, Machine Learning, Machine Learning Algorithms, Decision Trees

1. Introduction

The hiring process is the set of steps that organizations go through to recruit, evaluate, and select new employees. It typically involves job posting and recruitment, resume screening and initial interview, testing and background checks, final interview, job offer, and acceptance. The process may vary depending on the organization's size, culture, and the type of position being filled, but the goal is to find the best candidate for the job and ensure a smooth transition into the company.

Machine learning (ML) is being increasingly used in the hiring process to help organizations make data-driven, objective hiring decisions. The use of this approach in hiring is growing in popularity due to its potential to improve accuracy, reduce bias, increase efficiency, save costs, and provide increased consistency in the hiring process. By analyzing large amounts of data, machine learning algorithms can identify patterns and relationships that may be difficult for human decision-makers to see. This can lead to more accurate predictions about candidate success and improve the overall quality of hires. Automating parts of the hiring process can reduce the need for manual labor and improve efficiency, while also reducing the opportunity for human biases and prejudices to influence the hiring process.

The goal of this article is to experiment with decision tree algorithms to predict which candidates should be hired or rejected. The effectiveness of each algorithm should be evaluated based on evaluation metrics. The algorithm with the highest efficiency should be chosen as the most effective for making hiring or rejection decisions.

COLINS-2023: 7th International Conference on Computational Linguistics and Intelligent Systems, April 20–21, 2023, Kharkiv, Ukraine
EMAIL: kyrylo.smelyakov@nure.ua (K. Smelyakov); yuliia.hurova.cpe@nure.ua (Y. Hurova); stiv161272@gmail.com (S. Osiievskyi)
ORCID: 0000-0001-9938-5489 (K. Smelyakov); 0000-0003-4608-613X (Y. Hurova); 0000-0003-0861-9417 (S. Osiievskyi)

2. Related works

As technology continues to advance, machine learning has become an increasingly popular tool across various industries. One area where machine learning has particularly excelled is in classification problems, due to its ability to learn from large amounts of data and make accurate predictions on new data. The articles [1, 2] provide a comprehensive overview of machine learning techniques and an introduction to the mathematical and statistical foundations of machine learning. They [1, 2] have played an important role in popularizing the use of machine learning for classification problems and have helped to establish machine learning as a fundamental tool in the field of data science. The paper [3] covers a wide range of issues regarding the analysis and classification of exchange segments of EEG that correspond to certain useful signals and artifacts. As well as solving the problems of the human identification, detection of victims of man-made disasters [4], the creation of modern search services [5], and e-learning systems [6, 7].

In recent years, the use of machine learning has gained popularity in the hiring process [2, 8] because companies are looking for ways to improve the efficiency and effectiveness of the hiring process. These works [2, 8] provide models for extracting personal data from external sources for joint analysis with data in resumes. For example, many big companies like IBM, Hilton, etc. have implemented machine learning algorithms in their hiring process to automate tasks and improve efficiency [9-11]. Including the protection of personal data, as described in [11]. On the other hand, the study [12] found that job applicants generally had low levels of trust and confidence in algorithmic decision-making in recruitment and selection processes. However, the study [12] also found that the applicants' perceptions varied according to their demographic and personal characteristics, with factors such as age, education, and prior experience influencing their attitudes toward algorithmic decision-making.

Machine learning algorithms can analyze large amounts of data [13], identify patterns and make predictions, potentially reducing time and costs. Including analysis photos [14, 15].

One study [16] conducted by the National Bureau of Economic Research found that machine learning algorithms can improve the quality of hires by up to 25%. The study used a dataset of over 300,000 job applicants and compared the performance of machine learning algorithms to traditional hiring methods. The results showed that the machine learning algorithms were able to identify top-performing candidates with greater accuracy, leading to improved quality of hires.

A company called Pymetrics has also developed a machine learning-based hiring platform that uses cognitive and emotional assessments to evaluate job candidates. The platform uses machine learning algorithms to analyze a candidate's responses to various tasks and games, which are designed to measure their cognitive and emotional traits. The platform has been used by several companies, including Unilever, to help identify candidates who possess the skills and traits needed for a particular role.

In the last few years, one of the most popular machine learning algorithms for solving both classification and regression problems is decision tree algorithms. The study [17] compares the performance resulting from the classification process of text documents using different machine learning algorithms. The decision tree shows a better measure of accuracy metrics. These algorithms have gained popularity due to their high accuracy, scalability, and ability to process large and complex datasets, create highly accurate models, and provide easily interpreted and understandable results. The articles [18-20] demonstrate the use of decision tree algorithms in various domains and show their potential for improving the accuracy, scalability, and interpretability of machine learning models.

Decision tree algorithms have shown promise in solving a variety of problems, including in the field of human resources. In the context of hiring, tree-based algorithms have been used to identify the most relevant features for predicting job performance and to build predictive models that can assist in the recruitment process. In the paper [21], three supervised classification algorithms are deployed to predict graduation rates from real data about undergraduate engineering students. The study [22] focuses on ways to support universities in admissions decision-making using data mining techniques to predict applicants' academic performance at the university. The paper [23] aims to present an effective method for predicting student employability based on the context and using Gradient Boosting classifiers. Predicting student employment is based on identifying the most predictive features affecting the hiring opportunity of graduates.

A company called HireVue has developed a predictive modeling tool [24] that uses a random forest algorithm to identify the most relevant features for predicting job performance and to build a predictive model that can assist in the recruitment process. The tool uses a combination of video interviews and psychometric assessments to predict job performance.

A study [25] by the University of Oklahoma used gradient boosting to predict the job performance of over 500 sales representatives in a financial services company with 81.5% accuracy. The algorithm outperformed traditional selection methods and identified important features such as work experience, communication skills, and personality traits, providing insights into successful characteristics.

One of the advantages of using tree-based algorithms in the hiring process is that they can handle both categorical and continuous data. This allows for a wide range of data types to be included in the predictive model, which can improve the accuracy of the predictions. Tree-based algorithms are also able to handle missing data, which can be a common issue in large datasets.

However, there are also some limitations to the use of tree-based algorithms in hiring. One limitation is that they are prone to overfitting, which can occur when the algorithm is too complex and fits the training data too closely. This can lead to poor generalization and reduced accuracy of new data. The article [26] presents suggested pruning strategies.

Despite these limitations, the use of tree-based algorithms in hiring offers the hope of improving predictive accuracy and reducing bias in the hiring process. As the technology continues to advance, it will be important for companies to explore the potential benefits of using these algorithms while also ensuring that they are used ethically and responsibly.

3. Methods and materials

The effectiveness of modern machine learning algorithms in hiring decisions depends on several factors, including the quality of the data used to train the algorithm, the choice of algorithm, and how the algorithm is integrated into the hiring process.

Consider the details of the dataset that will be used for the methods and experiments, the techniques proposed to solve the problem, and the metrics that will be used to evaluate the methods and select the most appropriate model.

3.1. Dataset description

The preparation of a dataset is a critical step in the process of predicting candidate suitability in the hiring process. It helps to ensure accurate predictions, avoid bias, and make more informed hiring decisions, leading to better outcomes for both the company and the candidates.

To solve a classification problem in hiring using machine learning algorithms, relevant data on job candidates needs to be collected and labeled appropriately. Here are some examples of data that can be used to solve this problem:

- Basic personal information about job candidates, such as their name, age, gender, and contact details;
- Data about educational backgrounds, such as their degree, major, and the institution where they studied;
- Data about previous work experience, such as the companies they worked for, their job titles, and their job responsibilities;
- Data about skills, such as programming languages, tools, and certifications;
- Data about candidates' performance in job interviews, such as their answers to specific questions or their overall impression;
- Data from standardized assessments that evaluate specific skills or traits relevant to the job.
- References that can include feedbacks and recommendations from previous employers or colleagues, which can provide insight into the candidate's work ethic, communication skills, and other important factors.

Once this data is collected, pre-processed, and cleaned up, it can be used to build a predictive model that will help identify candidates most likely to be a good fit for a particular position.

The found dataset [27] includes various attributes of 614 candidates. Attributes such as gender, work experience, education, internship, score, salary, offer history, location, and recruitment status. Possible values and detailed descriptions of all these attributes are given in Table 1.

Table 1
Description of attributes with possible values from the example dataset

Attribute	Description	Possible value	Usages
Serial_no	Person's identification number	1 – 614	–
Gender	Identity	Male/Female/Others	+
Python_exp	Experience in the Python programming language	Yes/No/Undefined	+
Experience_Years	Work experience	0 – 3	+
Education	Educational background	Graduated/Not Graduated	+
Internship	Internship experience	Yes/No/Undefined	+
Score	Score	0 – 700	–
Salary*10E4	Salary	150 – 81k	–
Offer_History	Availability of offers	Yes/No/Undefined	+
Location	Location	L1/L2/Others	–
Recruitment_Status	Hiring decision	Y/N	+

From Table 1, we can see that not all attributes are useful for the experiment and therefore can be excluded. Attributes such as serial no, score, salary, and location do not affect the prediction.

The result attribute is recruitment status. It's important to ensure that the data used to train the algorithm is not biased. If the data used to train the algorithm has biases, the algorithm may perpetuate and amplify these biases in its predictions. For instance, if the training data has a skewed representation of one specific group, the algorithm may be more likely to recommend candidates from that group, even if they are not the most qualified for the role. This can result in unfair and discriminatory hiring decisions that adversely affect qualified candidates from underrepresented groups. Therefore, it's crucial to identify and address any potential biases in the data used for training machine learning algorithms to ensure fair and equitable hiring practices. The quantitative characteristics of the example dataset by the resulting attribute (recruitment status) are shown in Table 2.

Table 2
Quantitative characteristics of the example dataset

Label	Value
Number of examples	614
Number of positive examples	422 (69%)
Number of negative examples	192 (31%)

From Table 2, we can that the number of positive examples is greater than the number of negative values. The dataset used in this study can be accessed via the following link [27].

3.2. Techniques

One of the many solutions in solving hiring problems is to use decision tree algorithms [28] because they are well suited for classification problems, which is a typical use case in the hiring process. Decision tree algorithms allow decision-makers to evaluate different criteria for each candidate. Each decision tree may have different criteria, such as skills, experience, availability, and communication style, which are ranked and compared to determine the best candidate.

There are the following most common decision tree algorithms [28] for building classification models:

- Decision tree;

- Random forest;
- Gradient boosted trees.

Consider the algorithms for solving the classification problems mentioned above.

3.2.1. Decision tree algorithm

A decision tree [29] is a popular machine learning technique for solving classification problems. It works by constructing a tree-based model of decisions and their possible consequences. Each node in the tree represents a feature or attribute, and each branch represents a possible value or outcome. The algorithm learns from the data and creates the best tree structure to classify new data points based on the previously seen data.

The process of creating a decision tree starts with the root node, which represents the most significant feature. The algorithm selects the feature that results in the greatest information gain or Gini index, a measure of impurity in the data, and splits the data into subsets based on the values of that feature. This process is repeated recursively for each subset until a stopping criterion is met, such as reaching a predefined depth or the data becoming too small.

When using decision tree algorithms to solve classification problems in hiring employees, the algorithm would be trained to learn from a set of labeled data representing previous successful and unsuccessful hires. The algorithm would then use this data to create a decision tree that can predict the likelihood of success for new job candidates based on a set of features.

One of the benefits of decision tree algorithms is their interpretability. Each node in the tree can be easily interpreted as a decision based on a specific feature, making it possible to understand how the algorithm arrived at its classification. Additionally, a decision tree can handle both categorical and continuous data and can even handle missing values.

However, a decision tree can also be prone to overfitting, which can reduce its accuracy on new data. To avoid overfitting, techniques such as pruning or merging multiple decision trees can be used.

In general, a decision tree is a powerful and interpretable algorithm for solving classification issues.

3.2.2. Random forest algorithm

Random forest algorithms [30] are a popular extension of the decision tree algorithm for solving classification problems. A random forest is an ensemble of decision trees, where each tree is trained on a different subset of the data and a different subset of the features. The final classification decision is made by combining the outputs of all the individual trees.

The random forest algorithm has several advantages over a single decision tree. First, it reduces the likelihood of overfitting by training multiple trees on different subsets of the data. This means that the algorithm can capture a wider range of patterns in the data, leading to better generalization performance on new data.

Second, random forests can handle large, high-dimensional datasets and noisy data. By randomly selecting subsets of features for each tree, the algorithm can effectively reduce the number of features considered, leading to faster training and improved accuracy.

Finally, random forests can be easily parallelized, allowing for efficient processing on large datasets.

The process of creating a random forest involves training multiple decision trees, each with a randomly selected subset of the training data and features. During training, each tree is grown independently, with no pruning or early stopping applied. The final classification decision is then made by combining the outputs of all the individual trees, typically by taking a majority vote.

One of the challenges of random forests is finding the optimal number of trees to use in the ensemble. Adding more trees to the forest can increase accuracy, but also increases the computational cost and can lead to overfitting. Cross-validation and other methods can be used to find the optimal number of trees for a given problem.

In summary, random forest algorithms are a powerful extension of the decision tree algorithm for solving classification problems. They offer several advantages over single decision trees, including reduced overfitting, improved accuracy on high-dimensional data, and efficient parallel processing.

3.2.3. Gradient boosted trees algorithm

Gradient boosting [31] is a machine learning technique used to build an ensemble of decision trees for solving classification problems. Gradient boosted trees (GBTs) are particularly effective when dealing with complex datasets that contain non-linear relationships between features and outcomes.

The goal of gradient boosting is to create a model that minimizes the overall error by combining the predictions of multiple weak learners (i.e., decision trees) sequentially. The algorithm starts by training a single decision tree on the entire dataset. The error between the predicted and actual outcomes is then calculated, and a second decision tree is trained on the residual error. The process is repeated for a pre-specified number of iterations, with each new tree trained on the residual error of the previous tree.

In GBTs, the contribution of each tree to the final prediction is weighted according to its accuracy, with more accurate trees given higher weights. This ensures that the final model gives more weight to the most accurate trees, improving its overall performance.

One of the key advantages of GBTs is that they can automatically process missing data and categorical features without the need for data preprocessing. This makes them particularly useful for dealing with complex real-world datasets, where preprocessing can be time-consuming and error-prone.

However, GBTs can be prone to overfitting, particularly when the number of trees in the ensemble is large. Regularization techniques, such as early stopping and shrinkage, can be used to prevent overfitting and improve the generalization performance of the model.

In summary, gradient boosted trees are a powerful and widely used technique for solving classification problems. They offer several advantages over other machine learning algorithms, including their ability to handle missing data and categorical features, and their high accuracy on complex datasets. However, it's important to ensure that the model is regularized to prevent overfitting and to ensure its generalization performance on new data.

3.3. Model evaluation metrics

To determine the quality of the decision tree partitioning, it is necessary to calculate the Gini impurity, entropy, or information gain for each feature in the dataset and select the feature that results in the highest gain as the splitting criterion for the current node. This process is repeated recursively for each child node until a stopping criterion is met, such as when all instances in a node belong to the same class, or when a maximum tree depth or a minimum number of instances per leaf is reached.

Gini impurity is a measure of how often a randomly chosen element from the dataset would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

$$Gini = 1 - \sum_{i=1}^{n} p_i^2, \quad (1)$$

where p_i – the probability of n element; n – the total number of classes.

Entropy is a measure of the amount of uncertainty in the dataset.

$$Entropy = -\sum_{i=1}^{n} p_i * log_2(p_i), \quad (2)$$

where p_i – the probability of n element; n – the total number of classes.

Calculating entropy requires the use of logarithms, which can be more computationally complex than calculating the Gini Index. As a result, calculating the Gini Index may be faster than calculating entropy.

Evaluating the effectiveness of a model involves measuring the model's performance on a set of evaluation metrics [32]. Several evaluation metrics can be used to evaluate the performance of a model, including accuracy, precision, recall, and F1 score.

The accuracy, recall, and precision metrics are calculated directly from the confusion matrix result and are percentage metrics based on the absolute number seen in the confusion matrix.

A confusion matrix is an n*n table used to describe the performance of a classification model. Each row of the confusion matrix represents the actual class, while each column represents the predicted class. There are only two categories of results in our classification – yes or no (1 or 0). A confusion

matrix is a combination of our prediction (1 or 0) and the actual value (1 or 0). The confusion matrix diagram of our classification model is described in Figure 1.

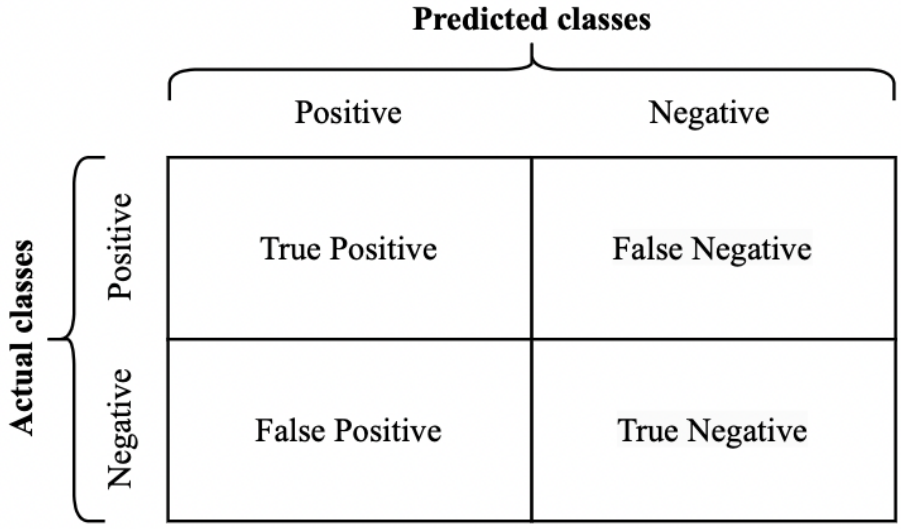

Figure 1: Confusion matrix for binary classification

A confusion matrix diagram is a useful tool for visualizing the performance of a binary classification model. A tool such as draw.io [33] was used to create this diagram. The diagram (Figure 1) can be found at the following link [34].

The recall is defined as the ratio of the number of true positives to the total number of actual positives.

$$Recall\ (R) = \frac{TP}{TP + FN}, \tag{3}$$

where TP – the number of positive examples correctly classified by the model; FN – the number of positive examples that were incorrectly classified as negative by the model.

A high recall score indicates that the model is effective at identifying positive examples, while a low recall score means that the model may be missing some positive examples.

The recall is particularly important in cases where missing a positive example can have serious consequences. However, a high recall score may come at the cost of lower precision, as the model may also identify some negative examples as positive.

Precision is defined as the ratio of the number of true positive results to the total number of true positive results.

$$Precision\ (P) = \frac{TP}{TP + FP}, \tag{4}$$

where TP – the number of positive examples correctly classified by the model; FP – or the number of negative examples that were incorrectly classified as positive by the model.

A high precision score indicates that the model is effective at identifying only the positive examples that are positive, while a low precision score means that the model is making many false positive predictions.

Precision is particularly important in cases where falsely identifying a negative example as positive can have serious consequences, such as in fraud detection or spam filtering. However, a high precision score may come at the cost of a lower recall, as the model may miss some actual positive examples. Therefore, the optimal balance between precision and recall will depend on the specific needs.

Accuracy is defined as the ratio of the number of correct predictions to the total number of predictions and represents the proportion of instances that the model correctly classified.

$$Accuracy\ (A) = \frac{TP + TN}{TP + TN + FP + FN}, \tag{5}$$

where TP – true positives; TN – true negatives; FP – false positives; FN – false negatives.

A high accuracy score indicates that the model effectively predicts both positive and negative cases, while a low accuracy score means that the model makes many incorrect predictions.

Accuracy is a useful metric when the classes in the dataset are balanced, meaning that the number of positive and negative examples is roughly equal. However, when the classes are imbalanced, accuracy can be misleading and other metrics like precision and recall may be more appropriate.

F1 score is the harmonic mean of precision and recall and is used to evaluate the overall performance of the model in correctly identifying positive instances while minimizing the number of false positives and false negatives.

$$F1\ score = 2\frac{R * P}{R + P}, \tag{6}$$

where R – recall value (1); P – precision value (2).

A high F1 score indicates that the model is effective at identifying positive instances while minimizing false positives and false negatives.

The F1 score is particularly useful when the classes in the dataset are imbalanced, meaning that one class has significantly more instances than the other. In this case, a high accuracy score can be misleading since the model may simply predict the majority class. F1 score, on the other hand, takes both precision and recall and provides a more balanced evaluation of the model's performance.

Also, it is often necessary to estimate the training time of a model. Estimating the training time is important for several reasons:

- Resource allocation;
- Performance optimization;
- Cost optimization.

After training the machine learning model on the training dataset, metrics (3) – (6) are computed on the test dataset. The sum of these metrics is used to assess the quality of the model. By analyzing these metrics, we can compare the performance of different algorithms and determine which ones are the most efficient.

In summary, evaluating the effectiveness of a built model involves a combination of quantitative and qualitative techniques. It is important to use multiple metrics and techniques to get a complete understanding of the model's performance and to identify areas for improvement.

4. Experiment

The main goal of the experiment is to train models on a selected dataset [27] using different decision tree algorithms, to analyze the effectiveness of using machine learning algorithms to determine whether a candidate should be hired or not. And to determine which of the decision tree algorithms is most effective in making hiring or rejection decisions. To control overtraining, the experiment is conducted on both the training and test datasets. The experiment includes preparation for the experiment, training, analysis, and comparison of models.

4.1. Experiment preparation

Apache Spark computing platform [35] was used for the experiment. It is a tool for efficiently processing large amounts of data in a parallel and distributed manner. Spark can be used to process data stored on many different computers in a network, allowing it to scale horizontally and handle extremely large datasets. It provides several built-in libraries for tasks such as machine learning, graph processing, and streaming, making it a versatile tool for a wide range of data processing tasks. Spark has a user-friendly API in several programming languages, including Scala, Python, and Java [36], allowing developers to easily use and integrate it into their existing data processing pipelines. It also integrates well with other big data tools, providing a comprehensive and integrated big data platform.

The Java programming language was used in the experiment. The Java programming language was chosen as the programming language because of its reliability, platform independence, and strong support for distributed computing, which makes it suitable for large-scale data processing with Spark.

The process of training a classification model using decision tree algorithms involved a combination of data preparation, data processing, algorithm selection, model training, evaluation, tuning, and deployment. Figure 2 summarizes the experimental workflow for the training phase, providing an

overview of the process. Each step is described in more detail below, providing a complete overview of the methodology.

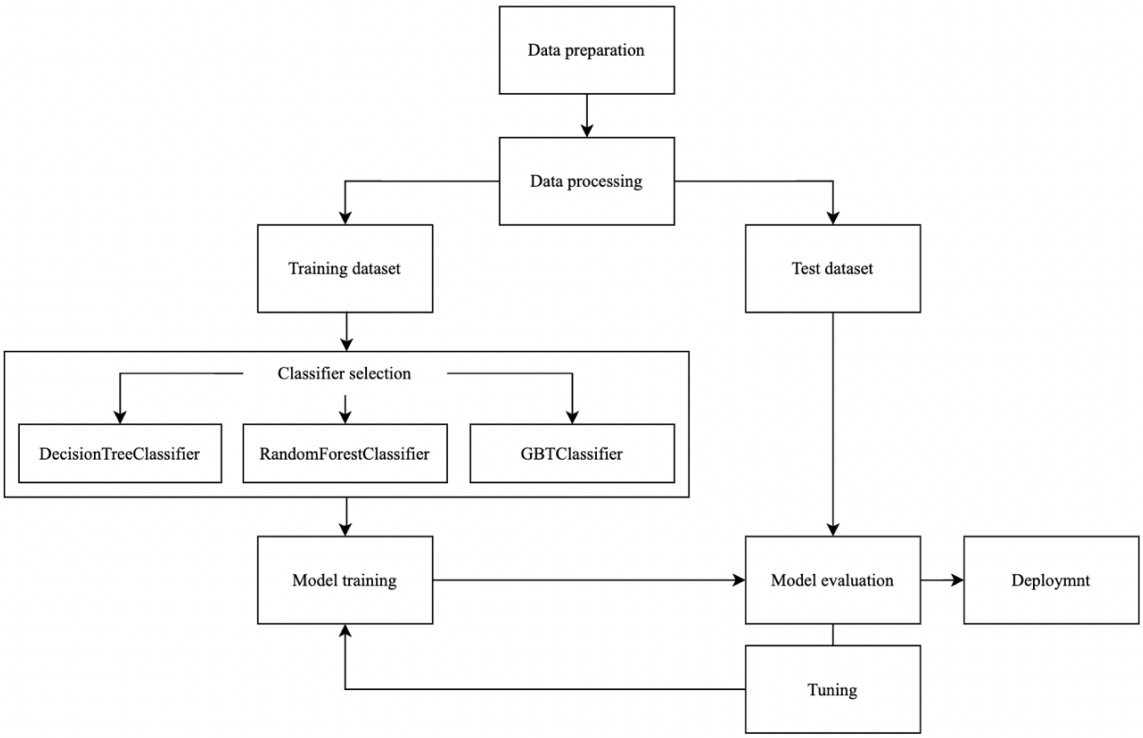

Figure 2: The experimental workflow

The diagram (Figure 2) was drawn using draw.io [33] and can be accessed via the link [37].

The first step is to prepare the data for the model by cleaning and processing it. Then, the necessary Spark libraries are imported and the data is loaded into a Spark DataFrame. Next, the data is pre-processed using the Spark machine learning API to scale, normalize, and encode categorical variables.

It is highly recommended to normalize the data before applying decision tree algorithms to ensure accurate and consistent results. Normalization of data is the process of bringing all attribute values into some desired range to ensure that each feature contributes equally to the analysis. Decision tree algorithms are sensitive to differences in the scales and distributions of data characteristics, which can lead to incorrect behavior of the algorithm and unreliable results. By normalizing the data, we can mitigate these problems and improve the accuracy and effectiveness of the decision tree algorithms.

Table 3 presents the non-normalized attributes of our dataset, including their initial values and corresponding normalized values.

Table 3
Non-normalized attributes with initial and normalized values

Attribute	Initial value	Normalized value
Gender	Male/Female/Others	1/0/2
Python_exp	Yes/No/Undefined	1/0/2
Education	Graduated/Not Graduated	1/0
Internship	Yes/No/Undefined	1/0/2
Recruitment_Status	Y/N	1/0

Before selecting the decision tree algorithm and training a model, it is important to divide the data into two sets – a training dataset and a test dataset. The training dataset is used to build the classification model, while the testing dataset is used to evaluate its performance. For the experiment, the dataset was divided into training and test datasets in the proportion of 80-20. The quantitative characteristics of the datasets are described in detail in Table 4.

Table 4
Quantitative characteristics of the training and test datasets

Label	Training data	Test data	Total
Example percentages (%)	80	20	100
Number of examples	498	116	614
Number of positive examples	343	79	422
Number of negative examples	155	37	192

After dividing the data into training and test datasets, the model can be trained on the training dataset. The model uses the data from the training dataset to learn how to make predictions. The recruitment status, which is defined by two groups – yes or no, was chosen as the result attribute of the dataset.

Once the data have been prepared, decision tree algorithms such as decision trees, random forests, or gradient boosted trees are selected. The selected classification model is then trained on the pre-processed data using Spark's MLlib library, specifying the model parameters and the training algorithm.

After the model has been trained, it can be used to make predictions on the test dataset. Predictions can be compared to the actual values on the test dataset to evaluate the effectiveness of the model. The effectiveness of the model is evaluated using metrics such as accuracy, precision, recall, and F1 score.

After analyzing the evaluation metrics, the model can be fine-tuned to improve its performance. This process may involve adjusting model parameters, selecting or removing features, and exploring different algorithms to find the most efficient ones.

The following describes several input parameters that were used to train the classification model using decision tree algorithms:

- MaxDepth controls the maximum depth of the tree (is used in all decision tree algorithms, the default value is 5);
- MaxBins controls the maximum number of bins used for discretizing continuous features (is used in all decision tree algorithms, the default value is 32);
- Impurity specifies the impurity measure used for tree building (the default value is "Gini" for decision trees and random forests);
- NumTrees is specific to random forests and controls the number of trees in the forest (the default value is 20);
- MaxIter is specific to gradient boosted trees and controls the maximum number of iterations in the model (the default value is 20);
- StepSize is specific to gradient boosted trees and controls the step size for each iteration of the model (the default value is 0.1).

It's important to note that the effectiveness of each algorithm is highly dependent on specific variables. Therefore, it is important to adjust these parameters to optimize model performance.

The steps of the experiment that concern the training of the model are described in more detail below.

4.2. Model training

Based on the proposed decision tree algorithms, software was developed to train the model using the selected algorithm and obtain the results of the experiment. Three different decision tree algorithms were chosen to train the models: decision tree, random forest, and gradient boosted trees. The current version (3.3.1) of Spark's MLlib library provides two types of impurity measures for the decision tree and random forest classifiers – Gini index and entropy. Therefore, based on these conditions, the following 5 models with appropriate parameters were trained for the experiment:

1. Decision tree model (impurity: Gini index, max depth: 5, max bins: 32, min instances per node: 1, min weight fraction per node: 0, min info gain: 0, max memory in MB: 256);
2. Decision tree model (impurity: entropy, max depth: 5, max bins: 32, min instances per node: 1, min weight fraction per node: 0, min info gain: 0, max memory in MB: 256);
3. Random forest model (impurity: Gini index, max depth: 5, max bins: 32, number of trees: 20, min instances per node: 1, min weight fraction per node: 0, min info gain: 0);

4. Random forest model (impurity: entropy, max depth: 5, max bins: 32, number of trees: 20, min instances per node: 1, min weight fraction per node: 0, min info gain: 0);
5. Gradient boosted trees model (max depth: 5, max bins: 32, min instances per node: 1, min weight fraction per node: 0, min info gain: 0, max memory in MB: 256, max iteration: 20, loss type: logistic, step size: 0.1, subsampling rate: 1).

An important step in training is the measurement of model training time. Training time is an important factor to consider when choosing an algorithm for data analysis because it can significantly affect the efficiency and speed of analysis. By understanding the training time of each algorithm, the most efficient and effective method of analysis can be selected, leading to reliable and timely decision-making. Table 5 shows the training time results for models trained using decision tree algorithms.

Table 5
Results of model training time

Model	Training time, ms
Decision tree (Gini index)	1846
Decision tree (entropy)	803
Random forest (Gini index)	964
Random forest (entropy)	789
Gradient boosted trees	1728

As shown in Table 5, the training time required for each algorithm differs significantly, with the random forest being the fastest, and the decision tree and gradient boosted trees taking longer.

After training the models, a series of experiments were conducted by randomly generating several training and test datasets, and the average values of the metrics were calculated. Several metrics, including training time, accuracy, precision, recall, and F1 score, were used to evaluate the performance of the classification model. The results of the experiments are given below.

5. Results

Consider the results of the experiment. The overall training time and average accuracy, precision, recall, and F1 score are presented below.

5.1. Results of the experiments with the decision tree model

The results of the experiments that were obtained by testing the decision tree model on the test dataset are presented in Tables 6 and 7. Table 6 shows the accuracy, precision, recall, and F1 score of the decision tree model trained using the Gini index criterion.

Table 6
The results of the decision tree model experiment using the Gini index

Metric	Value
Training time, ms	1846
Accuracy, %	0.7379
Precision, %	0.89
Recall, %	0.2352
F1 score, %	0.6817

As shown in Table 6, the training time for the decision tree model is 1.846 seconds. The model achieved an overall accuracy of 73.8%, which means that it was able to correctly predict the class labels for 73.8% of the samples in the test dataset. Moreover, the model achieved a precision of 89% and recall of 23.5%, which means that it was able to correctly identify 89% of the positive samples and avoid false

positives. Also, our model achieved an F1 score of 68.2%, which indicates that it is reasonably good at both precision and recall.

Overall, the decision tree model achieved a reasonable accuracy and F1 score, with a high precision but a relatively low recall. This suggests that the model is good at correctly identifying positive cases but may have missed a significant number of positive cases in the dataset.

Table 7 shows the results of the evaluated metrics for the decision tree model trained using the entropy criterion.

Table 7
The results of the decision tree model experiment using entropy

Metric	Value
Training time, ms	803
Accuracy, %	0.7475
Precision, %	1
Recall, %	0.2353
F1 score, %	0.6895

As shown in Table 7, the training time for the model is 0.803 seconds. The model achieved an overall accuracy of 74.75%, with precision and recall scores of 100% and 23.5%, respectively, and an F1 score of 68.95%.

Overall, the results suggest that the model is relatively accurate but struggles with identifying positive cases. The high precision score indicates that the model is good at identifying true positives, but the low recall score suggests that it may be missing a significant number of actual positive cases.

The results of the efficiency evaluation of our decision tree model using both the Gini index and the entropy criterion show very similar results, with slightly better results obtained using the entropy criterion.

Overall, the experiment using the decision tree algorithm was successful in producing a model.

5.2. Results of the experiments with the random forest model

The results of the experiments that were obtained by testing the random forest model on the test dataset are presented in Tables 8 and 9. Table 8 shows the results of the evaluated metrics for the random forest model trained using the Gini index criterion.

Table 8
The results of the random forest model experiment using the Gini index

Metric	Value
Training time, ms	789
Accuracy, %	0.7766
Precision, %	1
Recall, %	0.3235
F1 score, %	0.7355

As shown in Table 8, the training time for the model is 0.789 seconds. The model achieved an overall accuracy of 77.7%, with precision and recall scores of 100% and 32.4%, respectively, and an F1 score of 73.6%.

The provided results indicate that the model is performing reasonably well, but it is having difficulty identifying positive cases accurately. Although the precision score is high, which means that the model is good at predicting true positives, the low recall score suggests that the model is missing a significant number of actual positive cases.

Table 9 shows the results of the evaluated metrics for the random forest model trained using the entropy criterion.

Table 9
The results of the random forest model experiment using entropy

Metric	Value
Training time, ms	964
Accuracy, %	0.7864
Precision, %	1
Recall, %	0.3529
F1 score, %	0.75

As shown in Table 9, the training time for the model is 0.964 seconds. The model achieved an overall accuracy of 78.6%, with precision and recall scores of 100% and 35.3%, respectively, and an F1 score of 75%.

Overall, the results suggest that the random forest model is relatively accurate but still struggles with identifying positive cases. The high precision score indicates that the model is good at identifying true positives, but the relatively low recall score suggests that it may still be missing a significant number of actual positive cases.

The results of the efficiency evaluation of our random forest model using both the Gini index and the entropy criterion also show very similar results, with slightly better results obtained using the entropy criterion.

Overall, the experiment using the random forest algorithm was successful in producing a model.

5.3. Results of the experiment with the gradient boosted trees model

Table 10 shows the results of the evaluated metrics for the trained gradient boosted trees model.

Table 10
The results of the gradient boosted trees model experiment

Metric	Value
Training time, ms	1728
Accuracy, %	0.7087
Precision, %	0.75
Recall, %	0.1764
F1 score, %	0.6417

As shown in Table 10, the training time for the model is 1.728 seconds. The model achieved an overall accuracy of 70.9%, with precision and recall scores of 75% and 17.6%, respectively, and an F1 score of 64.2%.

In summary, the model achieved a moderate level of accuracy, but it is having difficulty identifying positive instances, as seen in the low recall score. The precision score is better, indicating that the model is good at identifying true positives.

Overall, the experiment using the gradient boosted trees algorithm was successful in producing a model.

6. Discussions

Tables 6-10 provide a detailed analysis of the experimental results obtained from the study, and it shows that the use of decision tree algorithms has demonstrated promising levels of efficiency in terms of accuracy, precision, and recall. The accuracy of the algorithms was high, indicating that the models were able to correctly classify a large proportion of the job candidates.

Before analyzing the efficiency of different decision tree algorithms, we evaluated the impact of impurity measures on our classification problem. The Gini index and entropy criterion are two common measures used to evaluate impurity in decision trees. A comparison of the two measures showed that

they had a minor impact on the efficiency of our tree models, with the entropy criterion slightly outperforming the Gini index. These findings suggest that either of these measures can be used effectively for our classification problem.

Based on the comparison of the classification models, the random forest algorithm is the most effective in terms of achieving higher overall accuracy, precision, recall, and F1 score, outperforming both decision tree and gradient boosted tree models in all these metrics. With an overall accuracy of 78.6%, the random forest model achieved the highest accuracy score among the three models. The precision and recall scores for the random forest model were also higher than those of the other two algorithms, with a precision score of 100% and a recall score of 35.3%. In comparison, the decision tree algorithm achieved a precision score of 100% and a recall score of 23.5%, while gradient boosted trees achieved a precision score of 75% and a recall score of 17.6%.

However, the training time required for each algorithm varies significantly, with the decision tree being the fastest and random forest and gradient boosted trees taking longer. While the decision tree algorithm is the fastest and took only 0.803 seconds to train the model, the random forest and gradient boosted trees algorithms took 0.964 seconds and 1.728 seconds, respectively.

Therefore, the choice of which algorithm to use depends on the specific needs. If training time is a critical factor and high effectiveness is not as important, the decision tree algorithm could be a suitable option. However, if the highest level of accuracy and performance is required, the random forest algorithm should be the preferred choice.

It is important to note that the gradient boosted trees algorithm can also be a good alternative to the random forest algorithm. For example, if the training time is not as critical but there is a need to prioritize precision over recall, then gradient boosted trees could be a better choice than random forests.

Although the results of these experiments show that the random forest algorithm is the most efficient among the three algorithms in the hiring process, it is also worth noting that the size and quality of the dataset used in the experiments may have affected the efficiency of the algorithms' results. The difference in the comparison of the results obtained in the three experiments can be considered negligible, so the use of any of the considered algorithms was successful in creating the model.

The result of this research provided valuable insight into how effective decision tree algorithms can be in helping to recruit and select the right candidates for a job. The best use of the models depends on the trade-off between training time, accuracy, precision, and recall, as well as the specific requirements. The decision tree algorithm can be a suitable option for applications where training time is critical, while the random forest algorithm should be preferred for applications where the highest level of accuracy and performance is required. The gradient boosted trees algorithm can be a good alternative to the random forest algorithm in some situations, depending on the specific needs.

Furthermore, this research provides a foundation for future research in this area. As decision tree algorithms continue to evolve and become more sophisticated, there is a need for further research to explore their effectiveness in different contexts and under different conditions. This could include investigating the impact of decision tree algorithms on diversity and inclusion in the hiring process, as well as their effectiveness in predicting long-term job performance.

Ultimately, the result of these experiments has practical implications for decision-makers involved in the recruitment and selection of job candidates. By providing evidence of the effectiveness of decision tree algorithms in this context, this research can help decision-makers make better hiring decisions and improve the overall quality of their workforce. Additionally, the study highlights the potential for further research in this area, which can continue to inform and improve the way organizations approach the hiring process.

7. Conclusions

This study identified various methods and algorithms that can be used when making decisions when hiring candidates for a team. Several experiments were conducted using different machine learning algorithms such as decision trees, random forests, and gradient boosted trees. The experiments were conducted on a dataset that was divided into a training (80%) and a test (20%) dataset. The training dataset was used to train the model, and the test dataset was used to evaluate and compare the models

that were trained by the above algorithms with each other. The effectiveness of the models was evaluated using several metrics such as processing time, accuracy, precision, recall, and F1 score.

The results of the experiments showed that the Gini index and entropy criterion had a minor impact on the efficiency of our tree models, with entropy slightly outperforming Gini. This suggests that either measure can be used effectively for our classification problem. The comparison of the classification models revealed that the random forest algorithm achieved the highest overall accuracy, precision, recall, and F1 score among the three algorithms tested. However, the training time required for each algorithm varies significantly, with the decision tree being the fastest, and random forest and gradient boosted trees taking longer.

In conclusion, it's also important to note that machine learning algorithms should not be used to make the final hiring decision on their own, but rather as one tool among many in the hiring process. Human judgment and expertise should always be a part of the decision-making process, as algorithms can only provide predictions based on the data they were trained on and may not be able to fully consider factors such as cultural fit or soft skills.

8. References

[1] L. Ruff et al., "A Unifying Review of Deep and Shallow Anomaly Detection," in Proceedings of the IEEE, vol. 109, no. 5, pp. 756-795, May 2021, doi: 10.1109/JPROC.2021.3052449.

[2] M. V. Todescato, J. Hilger and G. Dal Bianco, "A New Strategy to Seed Selection for the High Recall Task," in IEEE Latin America Transactions, vol. 19, no. 12, pp. 2105-2112, Dec. 2021, doi: 10.1109/TLA.2021.9480153.

[3] M. M. Eduardo Vasconcellos et al., "Siamese Convolutional Neural Network for Heartbeat Classification Using Limited 12-Lead ECG Datasets," in IEEE Access, vol. 11, pp. 5365-5376, 2023, doi: 10.1109/ACCESS.2023.3236189.

[4] G. Krivoulya, I. Ilina, V. Tokariev and V. Shcherbak, "Mathematical Model for Finding Probability of Detecting Victims of Man-Made Disasters Using Distributed Computer System with Reconfigurable Structure and Programmable Logic," 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2020, pp. 573-576, doi: 10.1109/PICST51311.2020.9467976.

[5] Sharonova, N., Kyrychenko, I., Gruzdo, I., Tereshchenko, G. "Generalized Semantic Analysis Algorithm of Natural Language Texts for Various Functional Style Types", in CEUR Workshop Proceedings, 2022, 3171, pp. 16-26.

[6] Sharonova, N., Kyrychenko, I., Tereshchenko, G., "Application of big data methods in E-learning systems", 2021 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), 2021. – CEUR-WS, 2021, ISSN 16130073. - Volume 2870, PP. 1302-1311.

[7] J. I. Sales and F. de Sousa Ramos, "Learning in a Hiring Logic and Optimal Contracts," in IEEE Access, vol. 9, pp. 154540-154552, 2021, doi: 10.1109/ACCESS.2021.3128039.

[8] V. S. Pendyala, N. Atrey, T. Aggarwal and S. Goyal, "Enhanced Algorithmic Job Matching based on a Comprehensive Candidate Profile using NLP and Machine Learning," 2022 IEEE Eighth International Conference on Big Data Computing Service and Applications (BigDataService), Newark, CA, USA, 2022, pp. 183-184, doi: 10.1109/BigDataService55688.2022.00040.

[9] V. Rathor, N. Kaur and R. Rautela, "Employee Hiring using Machine Learning," 2022 International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2022, pp. 1-3, doi: 10.1109/ICCR56254.2022.9995882.

[10] T. Chakraborti, A. Patra and J. A. Noble, "Contrastive Fairness in Machine Learning," in IEEE Letters of the Computer Society, vol. 3, no. 2, pp. 38-41, 1 July-Dec. 2020.

[11] J. Lu, H. Liu, Z. Zhang, J. Wang, S. K. Goudos and S. Wan, "Toward Fairness-Aware Time-Sensitive Asynchronous Federated Learning for Critical Energy Infrastructure," in IEEE Transactions on Industrial Informatics, vol. 18, no. 5, pp. 3462-3472, May 2022, doi: 10.1109/TII.2021.3117861.

[12] A. Vos, K. Poels and A. Jacobs, "Exploring the Impact of Algorithmic Decision-Making on Recruitment and Selection Processes: The Perceptions of Job Applicants," in IEEE Transactions on Human-Machine Systems, vol. 51, no. 1, pp. 42-50, Feb. 2021.

[13] K. Smelyakov, A. Chupryna, D. Sandrkin and M. Kolisnyk, "Search by Image Engine for Big Data Warehouse," 2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 2020, pp. 1-4, doi: 10.1109/eStream50540.2020.9108782.
[14] K. Smelyakov, A. Chupryna, O. Bohomolov and I. Ruban, "The Neural Network Technologies Effectiveness for Face Detection," 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), 2020, pp. 201-205, doi: 10.1109/DSMP47368.2020.9204049.
[15] K. Smelyakov, A. Chupryna, O. Bohomolov and N. Hunko, "The Neural Network Models Effectiveness for Face Detection and Face Recognition," 2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), 2021, pp. 1-7, doi: 10.1109/eStream53087.2021.9431476.
[16] M. J. Barber, H. Kim, "Can machine learning improve the quality of hires? Evidence from a randomized controlled trial," National Bureau of Economic Research, Paper 27547, Jul. 2020.
[17] G. N. Awaludin et al., "Comparison of Decision Tree C4.5 Algorithm with K-Nearest Neighbor (KNN) Algorithm in Hadith Classification," 2020 6th International Conference on Computing Engineering and Design (ICCED), Sukabumi, Indonesia, 2020, pp. 1-6.
[18] R. Mukherjee and A. De, "Development of an Ensemble Decision Tree-Based Power System Dynamic Security State Predictor," in IEEE Systems Journal, vol. 14, no. 3, pp. 3836-3843, Sept. 2020, doi: 10.1109/JSYST.2020.2978504.
[19] C. -H. Hsu, "Optimal Decision Tree for Cycle Time Prediction and Allowance Determination," in IEEE Access, vol. 9, pp. 41334-41343, 2021, doi: 10.1109/ACCESS.2021.3065391.
[20] X. Wang and F. Liu, "Data-Driven Relay Selection for Physical-Layer Security: A Decision Tree Approach," in IEEE Access, vol. 8, pp. 12105-12116, 2020, doi: 10.1109/ACCESS.2020.2965963.
[21] Y. Nieto, V. Gacía-Díaz, C. Montenegro, C. C. González and R. González Crespo, "Usage of Machine Learning for Strategic Decision Making at Higher Educational Institutions," in IEEE Access, vol. 7, pp. 75007-75017, 2019, doi: 10.1109/ACCESS.2019.2919343.
[22] H. A. Mengash, "Using Data Mining Techniques to Predict Student Performance to Support Decision Making in University Admission Systems," in IEEE Access, vol. 8, pp. 55462-55470, 2020, doi: 10.1109/ACCESS.2020.2981905.
[23] O. Saidani, L. J. Menzli, A. Ksibi, N. Alturki and A. S. Alluhaidan, "Predicting Student Employability Through the Internship Context Using Gradient Boosting Models," in IEEE Access, vol. 10, pp. 46472-46489, 2022, doi: 10.1109/ACCESS.2022.3170421.
[24] HireVue. (n.d.). Our science. Retrieved from https://www.hirevue.com/our-science.
[25] P. O. Fernandes, J. Cunha, and J. Neves, "Predicting sales representative job performance using gradient boosting," Expert Systems with Applications, vol. 107, pp. 240-250, 2018.
[26] I. Yildirim and M. Celik, "An Efficient Tree-Based Algorithm for Mining High Average-Utility Itemset," in IEEE Access, vol. 7, pp. 144245-144263, 2019, doi: 10.1109/ACCESS.2019.2945840.
[27] Recruitment data. URL: https://www.kaggle.com/datasets/rafunlearnhub/recruitment-data.
[28] The Guide to Decision Tree-based Algorithms in Machine Learning (Including Real Examples). URL: https://omdena.com/blog/decision-tree-based-algorithms.
[29] Decision Tree Classification Algorithm. URL: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.
[30] Random Forest Algorithm. URL: https://www.javatpoint.com/machine-learning-random-forest-algorithm.
[31] Gradient Boosting Algorithm: URL: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/.
[32] How to Evaluate Classification Models. URL: https://www.edlitera.com/blog/posts/evaluating-classification-models.
[33] Draw.io. [Online]. Available. URL: https://app.diagrams.net.
[34] The confusion matrix diagram of the binary classification model. URL: https://drive.google.com/file/d/1IR81h_KrxTtRZkwX-zIl4dDABsaJp0IP/view?usp=sharing.
[35] Apache Spark Benefits. URL: https://www.ksolves.com/blog/big-data/spark/apache-spark-benefits-reasons-why-enterprises-are-moving-to-this-data-engineering-tool.
[36] Spark documentation. URL: https://spark.apache.org/docs/latest/api/java.
[37] The experimental workflow. URL: https://drive.google.com/file/d/1PCEkscE0ZaMJXwyYZh_adWr3FrciEtaw/view?usp=sharing.