I.J. Information Engineering and Electronic Business, **2025, 5, 93-146**
Published Online on October 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2025.05.07

Information Engineering for Fake Job Postings Classification in Electronic Business Based on Machine Learning Technology

Markiian-Mykhailo Paprotskyi
Department of Information Systems and Networks, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: markiian-mykhailo.paprotskyi.sa.2021@lpnu.ua
ORCID iD: https://orcid.org/0009-0002-5292-8559

Victoria Vysotska
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: Victoria.A.Vysotska@lpnu.ua
ORCID iD: https://orcid.org/0000-0001-6417-3689

Lyubomyr Chyrun
Ivan Franko National University of Lviv, Lviv, 79000, Ukraine
E-mail: Lyubomyr.Chyrun@lnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-9448-1751

Yuriy Ushenko*
Department of Computer Science, Educational and Research Institute of Physical, Technical and Computer Sciences, Yuriy Fedkovych Chernivtsi National University, 58012, Ukraine
E-mail: y.ushenko@chnu.edu.ua
ORCID iD: https://orcid.org/0009-0002-7620-5355
*Corresponding Author

Zhengbing Hu
School of Computer Science, Hubei University of Technology, Wuhan, China
E-mail: drzbhu@gmail.com
ORCID iD: https://orcid.org/0000-0002-6140-3351

Dmytro Uhryn
Department of Computer Science, Educational and Research Institute of Physical, Technical and Computer Sciences, Yuriy Fedkovych Chernivtsi National University, 58012, Ukraine
E-mail: d.ugryn@chnu.edu.ua
ORCID iD: https://orcid.org/ 0000-0003-4858-4511

Received: 06 March, 2025; Revised: 11 May, 2025; Accepted: 16 June, 2025; Published: 08 October, 2025

Abstract: This study investigates the application of machine learning methods for the classification of fraudulent job postings in e-business platforms. Using the publicly available fake_job_postings.csv dataset, textual and categorical features of vacancies were processed and vectorised through TF-IDF, HashingVectorizer, and optimised TF-IDF. Eight machine learning algorithms were compared, including Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, Multinomial Naive Bayes, Linear SVC, K-Nearest Neighbours, and XGBoost. The experiments demonstrate that XGBoost achieved the best performance (Accuracy = 0.990, Precision = 0.982, Recall = 0.998, F1 = 0.990) across all feature representations. Its superior results can be attributed to the ability of boosted ensembles to capture complex non-linear relationships in high-dimensional feature spaces while maintaining robustness against noise and class imbalance.

However, it should be noted that the evaluation was performed on a single static dataset. While the high recall shows the model's ability to reliably detect fraudulent ads in this context, questions remain about its generalisability. Fraud tactics evolve rapidly, and new job scams may significantly differ from patterns in the training data. This creates

a potential risk of overfitting to dataset-specific features, which limits direct transfer to real-world scenarios without continuous retraining and monitoring. The practical contribution of the study is a reproducible framework that integrates text and categorical processing, vectorisation, hyperparameter optimisation, and comparative model benchmarking. Such a framework could be embedded into online job platforms to support automated filtering of suspicious ads. Still, its deployment requires additional measures: periodic retraining with updated data, integration with platform APIs, and the inclusion of explainability modules to ensure transparency and user trust. Overall, the research demonstrates that ensemble-based models, particularly XGBoost, offer strong potential for fraud detection in the e-business labour market. At the same time, further work is necessary to validate model robustness on unseen and evolving fraudulent job posting strategies, ensuring scalability and reliability in production environments.

Index Terms: Machine Learning; Natural Language Processing; E-Business; Fake Jobs; Classification; Logistic Regression; Accuracy; Fullness; F1-Measure; Information Engineering

1. Introduction

In today's digital environment, online recruiting has become the main channel for job search, but the proliferation of fraudulent vacancies accompanies its growing popularity. Fake job postings are used to collect personal data, commit fraud, or engage victims in dangerous schemes. Previous research shows that detecting such ads is a difficult task due to the high similarity of fraudulent vacancies to legal ones and the dynamism of fraudulent tactics [1-4]. Despite the availability of works using classical machine learning algorithms (Logistic Regression, Naïve Bayes, SVM), ensemble methods (Random Forest, Gradient Boosting) and modern neural networks (BiLSTM, BERT), there are still several problems [5-14]. Firstly, most models are built and tested on static datasets, which do not guarantee their generalizability to new types of fraudulent ads. Secondly, many studies do not pay enough attention to the integration of textual and categorical features, although their combination can significantly improve the quality of classification. Thirdly, there is a lack of studies that systematically analyse the impact of various methods of text vectorisation (e.g., TF-IDF, Hashing, n-grams) on the performance of models. Thus, the scientific gap lies in the lack of a comprehensive analysis of the impact of different methods of presenting features (textual and categorical) and their combination with various machine learning algorithms for the task of detecting fraudulent vacancies. Its elimination is the goal of this work.

The rapid development of information technologies and the globalisation of the economy have contributed to the active spread of e-business, in which digital platforms have become key intermediaries between employers and job seekers. Today, millions of users interact with online employment resources every day, from international platforms like LinkedIn, Indeed, and Glassdoor to regional services and specialised portals. However, the increase in the availability of such platforms is accompanied by an increase in risks, among which one of the most acute problems is the spread of fake vacancies. Fraudulent job offers negatively affect not only individual users, causing financial and psychological losses, but also undermine trust in the electronic labour market as a whole. On a global scale, this phenomenon is becoming more and more relevant: according to research, the number of fake vacancies is growing in all regions of the world, and in the context of military conflicts, economic instability and internal migration, the risks for job seekers increase significantly. Young professionals and people who are urgently looking for work remain a particularly vulnerable group, as they are less aware of the signs of fraudulent schemes. In the context of today's e-business needs, it is vital to provide efficient, scalable, and intelligent tools to automatically detect fake jobs. The use of machine learning and natural language processing (NLP) technologies opens up opportunities for the creation of highly accurate systems capable of analysing both textual job descriptions and related structured features (industry, type of employment, presence of a logo, questions to the candidate, etc.). This approach allows increased security for users and the protection of their data, as well as the maintenance of the stability and reputation of digital platforms. Today, a significant part of communications and business processes has moved online, which has led to an increase in the number of fraud cases, particularly in the field of employment. Online job search resources, such as LinkedIn, Indeed, or Ukrainian counterparts, are increasingly becoming a platform for posting fake vacancies, the purpose of which is to collect personal data, financial gain, or social engineering. This issue is especially acute for Ukraine in the context of war, internal migration and unemployment. Therefore, the development of tools for automated detection of fraudulent vacancies is highly relevant. Compared to existing systems, which often rely on manual moderation or basic keyword searches, the proposed approach provides a more reliable, scalable, and intelligent solution that can be integrated into extensive job services or government digital systems.

When studying similar works and publications, automatic detection of fake vacancies formed several areas: deep neural networks for text, classic ML models with feature engineering, and applied open-source solutions. Here is a concise, but fact-rich review of such scientific papers, consistent with our topic (detection of fake vacancies) and results (strong classic models on TF-IDF/Hashing and stable leaders like XGBoost/ensembles). Here are different directions: classic ML on EMSCAD/Kaggle, deep networks, transformers (BERT approaches), and applied/review articles. The sources are from recent years and are often cited in the field.

The purpose of the study is to develop and evaluate an approach to automated detection of fraudulent vacancies based on machine learning, taking into account both textual and categorical features, as well as to evaluate the effectiveness of various vectorisation methods. To achieve this goal, the following tasks have been defined:

1. Analyse the available approaches to identifying fraudulent vacancies.
2. Perform preliminary processing of the data set, highlight text and categorical features.
3. Implement various methods of text vectorisation (TF-IDF, Hashing, optimised TF-IDF).
4. Build and compare machine learning models (Logistic Regression, Naïve Bayes, SVM, Random Forest, Gradient Boosting, XGBoost).
5. Optimise hyperparameters and evaluate results using cross-validation.
6. Analyse the strengths and weaknesses of the models, identify the risks of overtraining and generalizability.

The object of the study is the processes of analysis and classification of vacancies posted on digital employment platforms. The subject of the survey is methods and means of identifying signs and patterns that are characteristic of fake vacancies by analysing text descriptions and metadata. As part of the study, for the first time, a full-fledged framework was implemented that combines multifactor data processing (text, categorical, and binary variables), various text field transformers (TfidfVectorizer, HashingVectorizer), a large number of models, including ensemble methods (Random Forest, Gradient Boosting, XGBoost), which take into account class imbalances, as well as automatic selection of hyperparameters using cross-validation (GridSearchCV).

The main declared novelty of the study lies in the systematic comparison of classical algorithms, ensemble methods and optimised vectorisation of TF-IDF in the task of detecting fraudulent vacancies based on the Kaggle dataset. The author's contribution also includes the combination of textual and categorical features into a single model, which allows you to improve the accuracy of classification compared to the use of textual features only.

This paper examines the development and implementation of an intelligent system for the automated classification of jobs as fake or legitimate. The proposed approach is based on modern methods of information engineering focused on the needs of e-business. It takes into account the peculiarities of the global and regional labour market. The use of a combination of text and categorical analysis, various vectorizers and machine learning models allows you to achieve high classification accuracy. It provides the ability to integrate the solution into scalable digital services.

The article is organised as follows. Chapter 2 provides a review of the literature and existing approaches to detecting fraudulent jobs. Section 3 describes the dataset used, methods of preprocessing and vectorisation of features. Chapter 4 presents the experimental methodology, optimisation parameters, and models. Chapter 5 presents the results of the comparison of the models and their discussion. Chapter 6 presents conclusions and outlines directions for further research.

2. Related Works

The problem of detecting fraudulent job postings has attracted significant attention in recent years as online recruitment platforms have become the dominant medium for connecting employers and candidates. Numerous studies have highlighted the risks associated with fake vacancies, including financial fraud, identity theft, and large-scale scams exploiting the anonymity of digital ecosystems [15-16]. The literature on this subject can be divided into several methodological directions: (1) classical machine learning approaches based on feature engineering, (2) ensemble-based models that improve stability and accuracy, (3) neural network and transformer-based methods, and (4) hybrid frameworks with a focus on practical integration and explainability.

Classical ML approaches. Early works demonstrated that classical algorithms such as Logistic Regression, Support Vector Machines (SVM), Naïve Bayes, and Random Forest could achieve high accuracy on structured datasets like the Employment Scam Aegean Dataset (EMSCAD) or the Kaggle *Fake Job Postings* corpus [17-18]. These studies emphasised the importance of effective text preprocessing (tokenisation, stopword removal, lemmatisation) and feature engineering. In particular, TF-IDF and Bag-of-Words (BoW) representations combined with categorical metadata (e.g., employment type, company logo, telecommuting flag) provided robust baselines with F1-scores often exceeding 0.90 [2]. The advantage of such models lies in their interpretability and relatively low computational cost, which makes them attractive for real-world deployment on online platforms. Most applied works as [2-3] rely on EMSCAD (Employment Scam Aegean Dataset) or the popular Kaggle dataset with ~18 thousand km. Job descriptions (approx. 800 fraudulent) combining text and metadata [5]. It allows you to compare classical and modern approaches using standard representations (BoW/TF-IDF/Hashing) and advanced features (EMSCAD and Kaggle fake_job_postings). A separate direction – multi-class/multi-stage staging – is not just a "fake/not fake", but a classification of types of fraud (identity theft, corporate identity theft, pyramid/MLM, etc.) [2]. Such work confirms the feasibility of a deeper taxonomy and focuses on the further development of your system (risk-scoring, explainability of features). Work [2] based on EMSCAD (classical ML methods with feature engineering) shows the competitiveness of gradient boosting and related ensembles: for linguistic and structural features (description length, POS tags, company profile, etc.), F1 is fixed at the level of ≈0.88 and emphasises the importance of intelligibility of features. This line of work sets a practical standard for classical models and motivates systematic comparison with different text representations (TF-IDF, Hashing). Separate

studies such as [3] describe multilayer DNNs (deep dense networks on EMSCAD) with an accuracy of about 93–98% (depending on the source), but without detailing the F1 measure and without deploying a complete application pipeline (from data to service). It makes it difficult to directly match with classical models on standard vectorizers. Public repositories [4] as open-source application solutions demonstrate stack models (logistic regression, random forests, gradient boosting) with a combination of text and binary/numeric features (e.g., "telecommuting", the presence of a logo), which emphasizes the practical integrability and transparency of the code, but rarely contains a system experimental protocol with a comparison of vectorizers and hyperparameters.

Ensemble methods. Subsequent research increasingly turned towards ensemble learning methods, particularly Gradient Boosting Machines (GBM) and XGBoost. These approaches consistently outperform linear models on high-dimensional, sparse TF-IDF features by capturing non-linear relationships and handling class imbalance more effectively [19]. For instance, works based on EMSCAD demonstrated that XGBoost achieved F1-scores above 0.95 and showed greater robustness across different feature extraction methods compared to Random Forest or SVM [20]. Moreover, ensemble methods benefit from built-in regularisation, which reduces overfitting risks, a critical property when dealing with imbalanced datasets where fraudulent cases represent less than 10% of the total corpus.

Neural networks and deep learning. With the rise of deep learning, researchers began to experiment with Recurrent Neural Networks (RNNs), particularly BiLSTMs, and later with transformer-based architectures such as BERT, DistilBERT, and domain-specific models like Fraud-BERT [1, 21-22]. In the paper [1], the authors applied deep models (NLP) based on BiLSTM to ad texts in combination with numerical characteristics; high accuracy (~98.7%) and AUC (~0.91) were reported. The approach demonstrates the potential of deep architectures. Still, the works do not pay attention to the issues of product integration (API, modularity) and detailed comparative assessment with basic vectorisation methods. These models leverage contextual embeddings to capture semantic nuances in job descriptions, outperforming classical TF-IDF-based approaches in terms of recall and overall robustness to linguistic variation. For example, BiLSTM architectures achieved up to 98% accuracy on benchmark datasets [8], while BERT-based models further improved generalisability across domains [23]. However, the main drawback remains the substantial computational requirements and the difficulty of integrating such models into lightweight, real-time monitoring systems.

Hybrid and applied frameworks. A number of studies have focused on hybrid systems that combine classical and modern methods to balance accuracy and practicality. For instance, combining TF-IDF vectorisation with ensemble classifiers and supplementing them with explainability techniques such as SHAP or LIME provides not only competitive accuracy but also transparency for end users and platform administrators [24]. Some works propose multi-stage classification schemes that not only identify whether a job ad is fake but also categorise the type of fraud (e.g., identity theft, pyramid schemes, corporate impersonation) [25]. Others emphasise the engineering perspective, developing modular pipelines with APIs for real-time integration into job platforms or government monitoring systems [26].

Challenges and open problems. Despite substantial progress, several key challenges remain. First, most models are trained and evaluated on static datasets such as EMSCAD or Kaggle's corpus. It raises concerns about domain shift and temporal drift, as fraudulent tactics evolve [27]. A model optimised on historical data may fail to detect novel scams designed with different patterns. Second, many studies report accuracy as the primary metric, which can be misleading in highly imbalanced scenarios; precision, recall, F1-score, and PR-AUC are more reliable indicators [28]. Third, explainability and user trust are often underexplored in technical papers, yet they are essential for deployment in sensitive environments like labour market regulation. Finally, scalability and periodic retraining are necessary to ensure the long-term applicability of any deployed system [29-37].

The literature shows that classical ML methods (TF-IDF + SVM/LogReg) remain strong baselines, while ensemble models (especially XGBoost) provide the most consistent performance gains on structured benchmarks. Deep learning and transformer-based methods show promise for capturing more subtle linguistic cues and improving recall, but they are resource-intensive. The combination of textual and categorical features consistently improves classification quality, highlighting the importance of hybrid feature engineering. Future work must focus on generalisability, adaptability to evolving fraud strategies, and integration of explainability and monitoring mechanisms to support real-world deployment.

Online recruiting is growing sharply, and scammers are quickly adapting (including with the help of generative AI, deepfake identities, and live video interviews). Regulators and the media have recorded a multi-year increase in losses and fraud cases, making automated job screening a priority for employment platforms [38-39]. The most used public set is the Employment Scam Aegean Dataset (EMSCAD) (~18 thousand vacancies, significant class imbalance, text and metadata), as well as derivatives of Kaggle (Real or Fake Job Posting), which contain description texts and categorical fields (telecommuting, has_company_logo, has_questions, industry, function, etc.). These sets have become the standard for initial method comparison, but have risks of pattern ageing and domain specificity [1, 18, 40]. The first works demonstrated that Naive Bayes, SVM, Logistic Regression, and Random Forest on TF-IDF/BoW give competitive results due to simplicity and resistance to "noise". Text features were often combined with metadata (logo, questions to the candidate, type of employment), which consistently improved accuracy. Review papers summarise the benefits of various algorithms and emphasise the importance of high-quality preprocessing, combating imbalance and selection of hyperparameters [6, 41]. A number of papers demonstrate classical machine learning (TF-IDF/Hashing and ensembles/linear models), in particular, that on well-prepared text features (TF-IDF/Hashing), Random Forest, SVM/Linear SVC, Logistic Regression, and other ensemble/linear approaches yield competitive results (often

Accuracy/F1 ≳ 0.95–0.98) [6-8]. It correlates with your conclusion about ensemble efficiency and quality stability between different vectorizers. Several empirical studies record an advantage of scaffolding/gradient ensembles on EMSCAD/Kaggle and emphasise the importance of text preprocessing and feature engineering. There are works [9-10] based on deep approaches (MLP/RNN) on top of classical representations, where multilayer perceptrons or recurrent networks (BiLSTM) are added on the same base. They usually improve Recall, but require more careful balancing of classes and computing resources; Sometimes the increase is achieved at the cost of more complex tuning and longer training. Further research shows that gradient ensembles (XGBoost, GBDT) systematically lead in EMSCAD/"fake job postings" due to their ability to model nonlinearities in high-dimensional spaces and work with sparse TF-IDF features. Adding explainability (e.g., SHAP) allows you to identify the most influential textual and organisational attributes, which is helpful for platform moderation [42]. A new wave of work [11-12] based on Transformers and Learning Transfer (BERT/DistilBERT/RoBERTa) shows strong results of BERT-like models, in particular specialised variants like Fraud-BERT, which are superior to traditional BoW/TF-IDFs in capturing context and domain nuances. At the same time, the authors recognise the increased computational cost and complexity of production integration. For practical services (API/online filtering), compromise options on DistilBERT or a combination of transformer with easy metaclassification are often offered. Your result on strong ensembles (XGBoost/forests) on TF-IDF agrees well with these findings: classical approaches remain an attractive base, and transformers are the way to further increase in quality if resources are available. There are a number of recent review/application articles [13-14] summarising progress: typical datasets (EMSCAD/Kaggle), the role of class balancing, ensemble preferences, and transformer perspectives. Some of the work adds engineering aspects (web application/API, production pipeline), which are directly connected to our information engineering perspective. Work with BiLSTM/RNN and, especially, with BERT/Roberta shows an increase in quality due to contextual coding, sometimes generating new datasets from multiple sources to reduce overfitting to classic benchmarks. Fraud-BERT and similar approaches emphasise the advantages of transfer training on unbalanced data, but draw attention to the requirements for computing resources and the need for regular additional training [1,11, 43]. In addition to TF-IDF n-grams (1–3), the use of metadata (presence of a logo/questions, type of employment, industry, role) is practised, sometimes simple stylometric indicators ("excessive capital letters", URLs/contacts in the body, template phrases, abnormal salary ranges). Publications emphasise that the combination of text and categorical features consistently raises F1/ROC-AUC [1, 7]. Due to the imbalance of classes, Accuracy can be misleading; it is recommended to report Precision/Recall/F1, ROC-AUC and, preferably, PR-AUC, and use stratified cross-validation. Some of the works additionally use an explainer (SHAP/LIME) to validate fraud indicators [13, 42]. A key open challenge is domain-time drift: new fraud tactics can be radically different from EMSCAD/Kaggle patterns. Modern works offer periodic additional training, time-based split, collection of "fresh" multi-source datasets, active training, and hybrid ML+rules pipelines with manual moderation of risk cases [43]. XGBoost/GBDT and BERT-class most often lead historical benchmarks; 2) The text and categorical features give a stable increase; 3) production requires explainability, drift monitoring, regular data/model updates, and platform integration tools (APIs, moderation queues, audit log) [42].

Scientific and review papers [44-56] consolidate approaches from classical ML to modern transformers and recommend expanding features outside the text and tracking data shifts. At the same time, there are works with transformer architectures to detect fake vacancies. Still, they have not yet become a de facto standard in product solutions due to the need for significant computing resources and careful adaptation. Our study systematically compares a wide range of models (LogReg, RF, GB, DT, Multinomial NB, Linear SVC, KNN, XGBoost) under three text presentation modes (TF-IDF, HashingVectorizer, TF-IDF with hyperparametric optimisation). It evaluates them according to Accuracy/Precision/Recall/F1/AUC. In real experiments, XGBoost (F1≈0.990; Accuracy≈0.99) is stable for all three vectorisation options; also, strong results have Linear SVC (up to F1≈0.978–0.976) and optimised Multinomial NB. It confirms the conclusions of the classical line of research on the competitiveness of ensembles and linear methods on qualitative vector representations of the text, and at the same time offers a practical, easily integrated stack with a transparent evaluation methodology. Thus, in the context of published approaches, your contribution is a complete, reproducible comparison of models and vectorizers with hyperparametric optimisation and multi-criteria estimation, where XGBoost on TF-IDF/Hashing/TF-IDF (opt.) proved to be the best in terms of the set of metrics. Linear SVC and Multinomial NB formed strong baselines – all of which are consistent with trends in the literature but provide a more applied, engineering framework for integration into electronic business services.

The purpose of the study is to develop an intelligent system for evaluating the effectiveness of machine learning models for automatic detection of fraudulent vacancies based on the analysis of their description and structured features.

The main objectives of the study:

1. Conduct a complete cycle of processing the dataset of vacancies, including cleaning, analysis and preparation of textual and categorical features.
2. Implement several text transformations options, including TfidfVectorizer and HashingVectorizer.
3. Build, compare, and optimise different classification models (Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, Multinomial Naive Bayes, Linear SVC, K-Nearest Neighbours, XGBoost).
4. Perform hyperparametric optimisation of models (via RandomSearchCV).
5. Evaluate models by metrics: accuracy, precision, recall, F1-score, AUC.
6. Analyse the results obtained and determine the optimal model for the problem.

3. Materials and Methods

In today's digital space, the number of online resources for posting vacancies is skyrocketing. However, in parallel with this, the problem of fraudulent job ads is spreading. Fake vacancies can lead to financial losses, time waste, and emotional burnout of applicants. This problem is especially relevant for young professionals and students who do not have enough experience to distinguish real offers from fraudulent ones. Scammers use various methods of disguise: forging the names of well-known companies, tempting salaries, minimum requirements for candidates, promises of quick employment, etc. Visually, fake vacancies are not much different from real ones, especially if the applicant does not know what signs to look for. In addition, manually checking each vacancy requires significant time and resources, making it impossible to scale it for large platforms. Despite the urgency of the problem, there is a lack of tools to automatically check vacancies for signs of fraud, mainly localised for the Ukrainian market.

Thus, the lack of adequate means of automatic classification of fake vacancies significantly complicates the process of protecting job seekers from fraud and unscrupulous employers. To solve this problem, it is proposed to develop an intelligent system for automatically classifying vacancies as fake or genuine based on machine learning and natural language processing (NLP) methods. The main component of the project will be a text classification model trained on the basis of existing datasets, which contain signs of genuine and fraudulent vacancies. The solution is based on the analysis of the content of text descriptions of vacancies, as well as related information (for example, company names, location, and salary expectations). To achieve high accuracy, a combination of machine learning models (e.g. Logistic Regression, Random Forest, XGBoost) and text vectorisation (TF-IDF, Hashing Vectorizer, Word Embeddings) will be used. The user will be able to integrate the system into real online platforms or use it locally to check suspicious ads. It will significantly reduce the risk of getting into fake vacancies, increase trust in online job search resources, and save time and effort for job seekers.

The general goal of the project is to create an intelligent system for automatic detection of fake vacancies based on the analysis of the text description of vacancies. The implementation of such a solution aims to reduce the risk of fraud when looking for a job, increase the safety of users of online resources with vacancies, and optimise the process of finding high-quality ads. The system will be helpful for job seekers, recruiting agencies, employment platforms, and developers of automated content monitoring systems. To specify the tasks of the project, a tree of goals has been built:

Domain research → *Data collection and processing* → *Machine learning models Integration* → *into a prototype* → *Testing and improvement*

At the first stage, it is necessary to conduct a comprehensive analysis of the subject area. It includes studying modern approaches to the classification of text data, analysing existing solutions for detecting fake vacancies, and evaluating algorithms and models used in such projects. The second key area is working with data. Search, cleaning, processing and study of datasets of fake vacancies will be carried out. Particular attention will be paid to identifying key features and correlations between textual and categorical characteristics of vacancies. The next step will be the development of machine learning models. It is planned to test several approaches to building models, including Logistic Regression, Random Forest, Gradient Boosting, XGBoost, SVM, Naive Bayes, using TF-IDF and other vectorisation methods. Hyperparameter optimisation will be carried out to improve quality. Next, the model will be integrated into the prototype of the system – a web interface or a console application for demonstrating work. The final stage will be testing the system, assessing the quality of the models (F1-score, accuracy, ROC AUC), comparing the results of different approaches and determining the most effective solution. It is also possible to add feedback collection functions for further training of the model. Morphological analysis was carried out to select the optimal architecture of the system.

Table 1. Main system parameters

Setting	Options (alternatives)
Input	Job Text, Metadata (Salary, Company, Country)
Model	Logistic Regression, Random Forest, Gradient Boosting, XGBoost, SVM, Naive Bayes, LSTM, Transformer
Vectorization method	TF-IDF, CountVectorizer, HashingVectorizer, Word2Vec
Teaching method	Complete training, additional training of the finished model
Data source	Kaggle (fake_job_postings.csv)
Result output method	Class (fake/real), probability, explanation of signs

The main alternatives are: Alt 1 (TF-IDF, Logistic Regression, complete training, Kaggle, class), Alt 2 (TF-IDF, Multinomial NB, full training, Kaggle, class) and Alt 3 (HashingVectorizer, Random Forest, complete training, Kaggle, probability). To justify the choice, the method of hierarchy analysis was used. The selection criteria included accuracy, speed of work, complexity of implementation, and the need for resources. After the calculations, the optimal alternative to Alt 2 was determined (TF-IDF, Multinomial NB, complete training, Kaggle, class). This configuration provides good accuracy and an acceptable level of complexity of implementation and integration into a user-friendly web interface.

The figure 1 shows a vector of alternative priorities (result of the hierarchical analysis method / AHP) that compares a set of system configurations (e.g., TF-IDF+LR, TF-IDF+NB, Hashing+RF) according to criteria (accuracy, speed, implementation complexity, resource requirements). Interpretation: higher vector weights mean a better combination of criteria; in the file, this is used to justify configuration selection (Alt2 – TF-IDF + Multinomial NB).

The UML/architecture figures (Fig.2–6) show the architectural solutions and help the reader reproduce the prototype implementation. A use case diagram is constructed to describe and depict the different use cases of the system and user roles. There are two main types of users in the system. User – interacts with the system through the command line (CLI), enters the text of the vacancy for classification, and receives the result. The diagram (Fig. 2) shows the capabilities of each type of user in the system. Classic UML-use-case: roles (user, administrator) and their scenarios (job entry, classification request, view results, moderation/recall) are shown. Serves to illustrate the expected interaction of the system with human users. Classes are not provided for in the implementation of the project, so approximate classes that would be used to create the project will be defined and specified. Preprocessor class – text cleaning, preparation for vectorisation. Vectorizer – implementation of TF-IDF transformation. FakeJobClassifier – Learn and use Multinomial Naive Bayes for classification. Evaluator – functions for evaluating accuracy, displaying metrics, and building a confusion matrix. Fig. 3 shows system class diagram (Preprocessor, Vectorizer, FakeJobClassifier, Evaluator, etc.). Explains the internal structure of the software implementation: which modules are responsible for text cleaning, vectorisation, model training, and evaluation. It helps to understand the interfaces and dependencies of the components. A sequence diagram is constructed to depict the sequence of actions during a single session of use. The diagram (Fig. 4) shows the scenario for classifying a vacancy:

User Typing → Preprocessor Clears Text Vectorizer → Converts Text to Vector TF-IDF → FakeJobClassifier Classifies Text → User Gets Reply

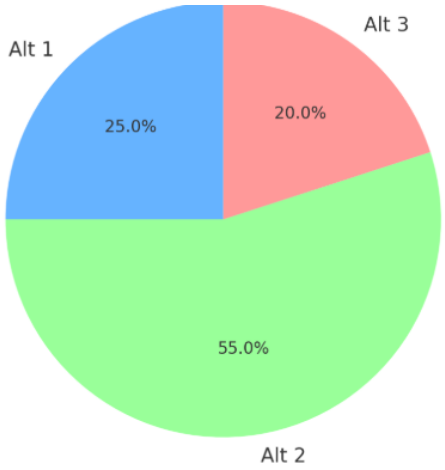

Fig. 1. Vector of priority of alternatives

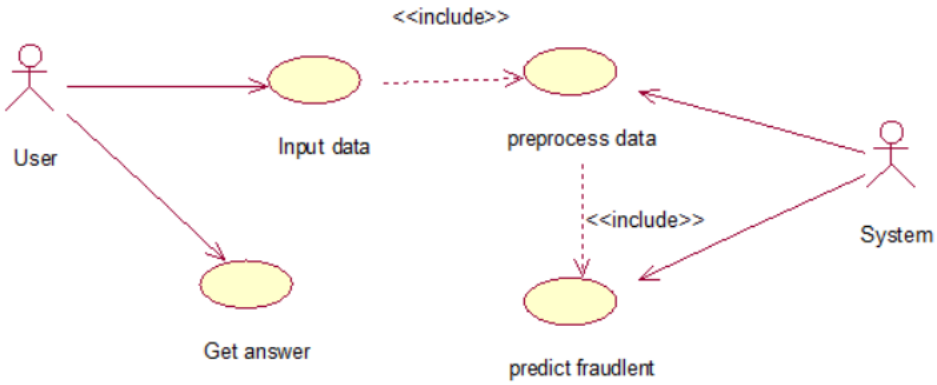

Fig. 2. Use case diagram

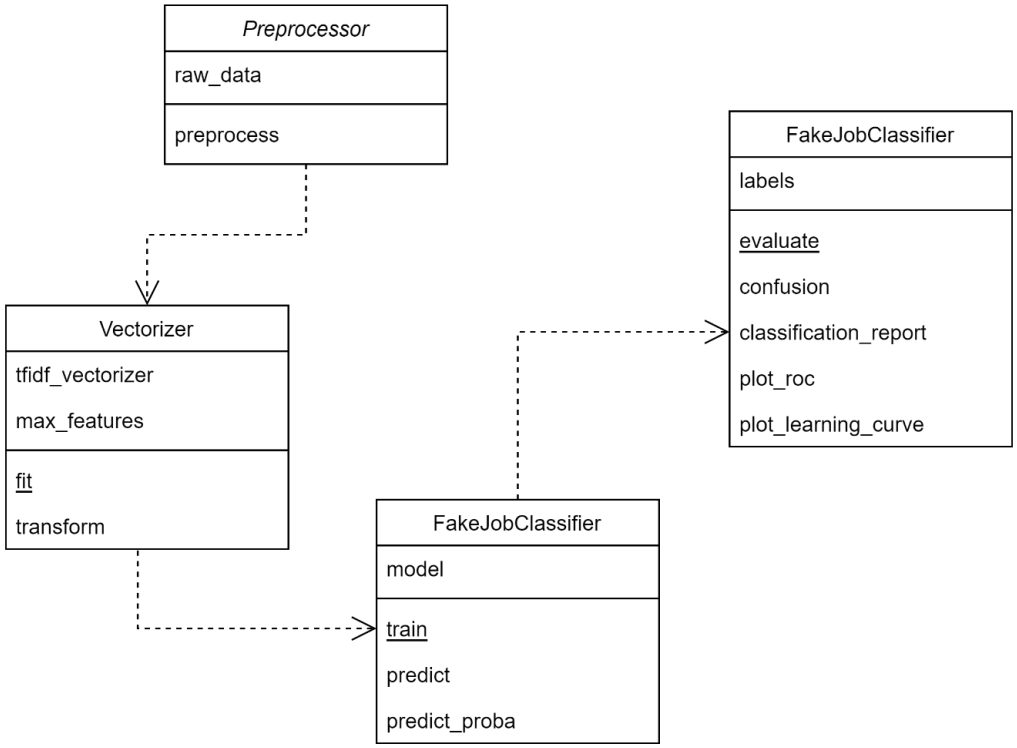

Fig. 3. Class Diagram

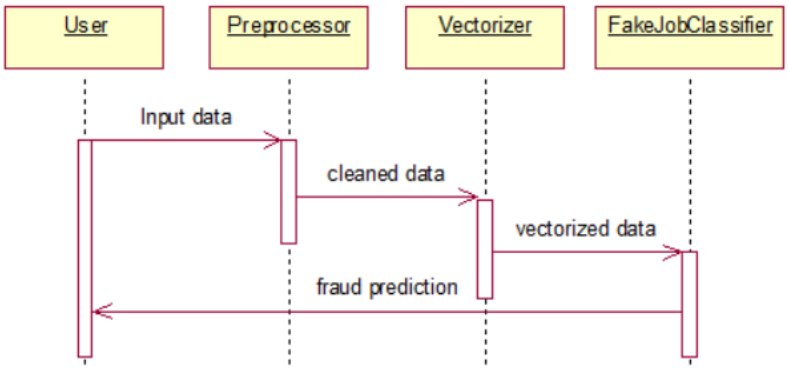

Fig. 4. Sequence diagram

The sequence of actions when processing a single request: user → preprocessor → vectorizer (TF-IDF) → classifier → response to the user. Illustrates the runtime scenario and the importance of the order of operations in the pipeline.

An activity diagram is built to demonstrate the system's operation. It shows the flow of operations:

Job Data Entry (User) → *Preprocessor* → *Convert Text to Vector TF-IDF* (Vectorizer) → *Text Classification* (FakeJobClassifier) → *Output (Fake / Real Vacancy)* (User)

To illustrate the architecture of the system, a diagram of components is built. The diagram (Fig. 5) shows the Processing head unit (uses main.py and evaluate.py), Main.py (contains the code responsible for all the main processes) and Evaluate.py (includes the code for evaluating the model). All components interact with each other, forming a single software system.

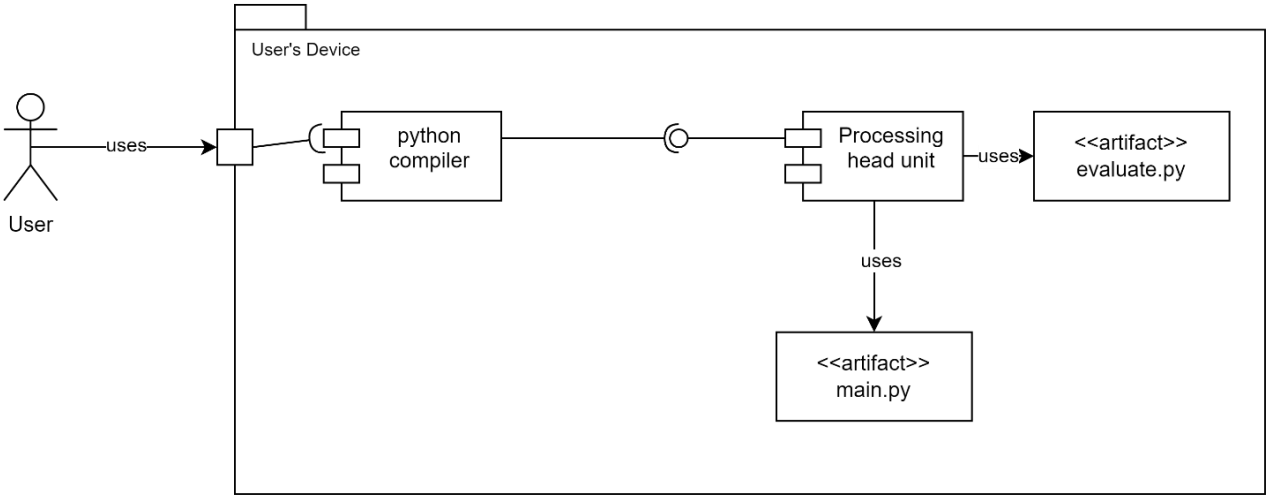

Fig. 5. Component Diagram

Component diagram of the software solution (main.py, evaluate.py, processing unit, etc.) and their interaction. Shows which scripts/modules perform the main stages (data processing, training, evaluation) and how they can be deployed as separate components.

A deployment diagram is built to demonstrate the overall configuration of the system. The system provides one deployment node – the user's local computer. Basic elements: Python compiler and main block, which performs all functions. Interaction occurs locally, through a Python compiler.

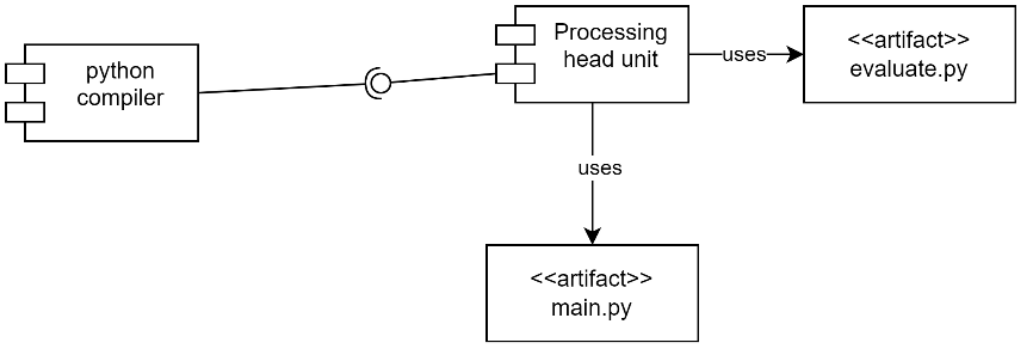

Fig. 6. Deployment Diagram

Deployment scheme: one node – the user's local computer with a Python interpreter. Explains the assumptions about the runtime environment (local demo/prototype) rather than the production cluster.

The study aims to develop and evaluate the effectiveness of machine learning models for the automated classification of vacancies into fake and legitimate. A set of fake_job_postings.csv containing textual, binary and categorical features is used as input. Target variable:

$$y = \begin{cases} 1, & \text{if the vacancy is fake,} \\ 0, & \text{if the vacancy is legitimate.} \end{cases}$$

The pipeline figures (Fig.7–9) document the methodology of the experiments and the location where TF-IDF optimisation is applied. Flowchart of the main research pipeline (Fig. 7): collection/EDA → preprocessing → vectorisation → hyperparameter search → model training → evaluation. The figure serves as an overview roadmap of experiments.

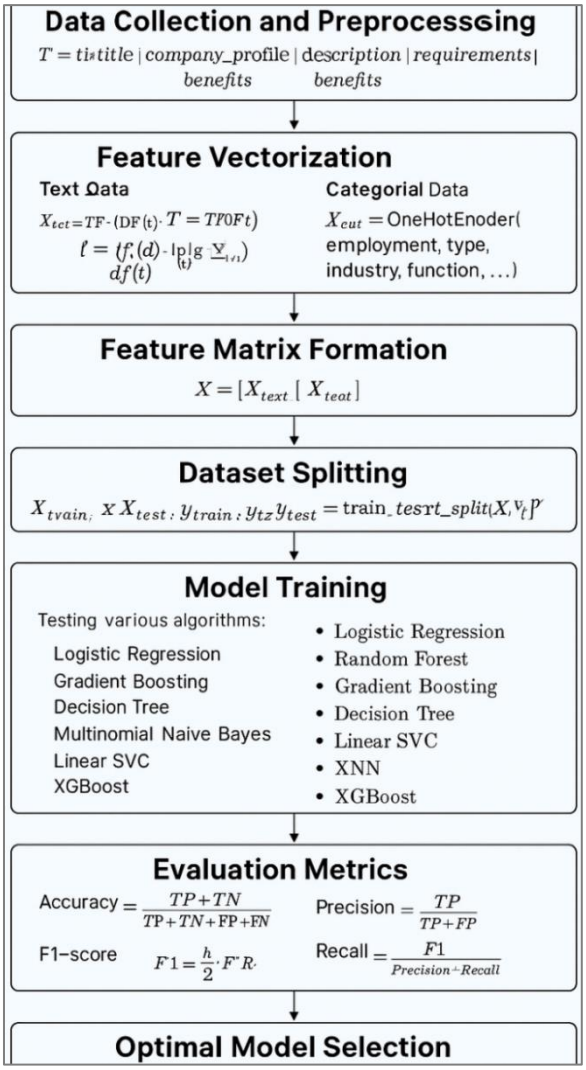

Fig. 7. Description of the main research pipeline

To solve the problem, a complete cycle of data processing, modelling and evaluation of the efficiency of algorithms is implemented. Research pipeline:

1. Data collection and pre-processing:
   - Merging Text Fields: $T = title \parallel company\_profile \parallel description \parallel requirements \parallel benefits$, where $\parallel$ means the concatenation of the strings.
   - Text Cleanup: $T' = \text{lower}(T) \setminus \{\text{numbers, punctuation, special characters}\}$.
2. Vectorisation of features:
   - For text data: $X_{\text{text}} = \text{TF-IDF}(T')$, $\text{TF-IDF}(t, d) = \text{tf}(t, d) \cdot \log \frac{N}{df(t)}$, where $tf(t, d)$ – is the frequency of the t term in the document d, $df(t)$ is the number of documents where t occurs, N – is the total number of records.
   - For categorical and binary features: $X_{cat} = OneHotEncoder(employment\_type, industry, function, \dots)$.
3. Formation of the final matrix of features: $X = [X_{text} \parallel X_{cat}]$.
4. Sample Separation: $X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}} = \text{train\_test\_split}(X, y, test\_size = 0.2, stratify = y)$.
5. Model training. Various algorithms were tested: Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, Multinomial Naive Bayes, Linear SVC, KNN, and XGBoost. Each model f was evaluated using parameter optimisation: $\theta^* = \arg\max_\theta \text{F1-score}(f_\theta(X_{\text{test}}), y_{\text{test}})$.
6. Evaluation metrics include Accuracy, Precision, Recall, and F1-score, also AUC: area under the ROC curve:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}, \text{Precision} = \frac{TP}{TP+FP}, \text{Recall} = \frac{TP}{TP+FN}, F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}.$$

7. The optimal model with the highest value of $F1$ in the test sample was selected for the final implementation.

Three feature extraction techniques were employed in this study: TF-IDF, Hashing Vectorizer, and optimised TF-IDF. Their selection was motivated by complementary strengths.

– TF-IDF remains one of the most widely used methods in text classification because it effectively balances term frequency with inverse document frequency, reducing the weight of common but uninformative words while highlighting discriminative terms. It provides a sparse but interpretable representation well-suited for linear classifiers.

– Hashing Vectorizer was included as a scalable alternative capable of handling extensive vocabularies without explicitly storing the feature index. Although it introduces the risk of hash collisions, it is computationally efficient and memory-friendly, making it relevant for real-world deployments on streaming data.

– Optimised TF-IDF extended the baseline TF-IDF by tuning parameters such as ngram_range, max_features, and min_df to maximise the discriminative power of the representation. This approach was chosen to test whether refined parameterisation yields performance gains over standard TF-IDF.

By evaluating models across these three vectorisation strategies, the study aimed to ensure that results were not artefacts of a single feature representation and to identify whether specific models benefit more from richer or more compact feature spaces.

Flowchart of the main research pipeline: collection/EDA → preprocessing → vectorisation → hyperparameter search → model training → evaluation. The figure serves as an overview roadmap of experiments.

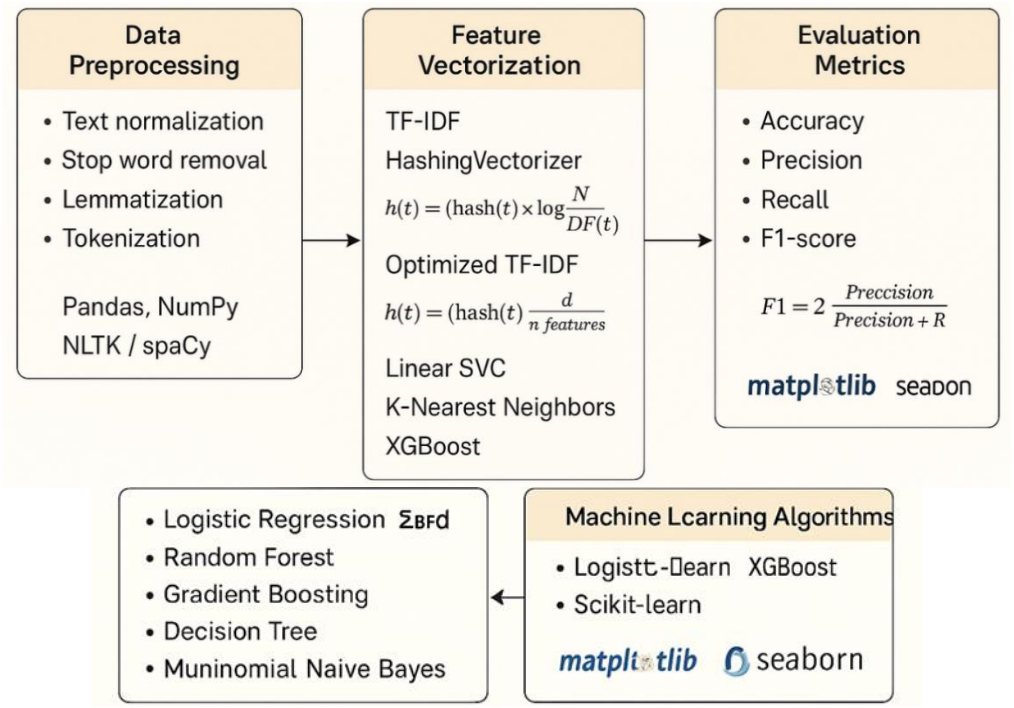

Fig. 8. Detailing the main pipeline of experiments

Research methods and tools:

1. Source and preparation of data:
   - Dataset: fake_job_postings.csv containing real and fraudulent jobs from various online resources.
   - Pre-treatment included:
     - Remove spaces in text and categorical fields.
     - Lowercase text reduction.
     - Remove stop words.
     - Lemmatisation and tokenisation.
     - Handling non-alphabetic characters and HTML tags.
     - Encoding categorical features into a numerical format (One-Hot Encoding).

2. Methods of vectorisation of features. The study included three approaches to converting text to numerical vectors:
   - **TF-IDF (Term Frequency – Inverse Document Frequency)**
     - A classic method that weighs the importance of a term depending on its frequency in the document and its rarity in the corpus.
     - The formula for the calculation is $TF - IDF(t,d) = TF(t,d) \times \log\frac{N}{DF(t)}$, where $TF(t,d)$ – is the frequency of the term t in the document d, $DF(t)$ is the number of documents in which the term t, N – is the total number of records.
   - **HashingVectorizer**
     - Converting text to a fixed number of features by hashing terms without saving a dictionary.

- The hash function is used: $h(t) = (hash(t)\ mod\ n\_features)$, which guarantees the constancy of the size of the vector regardless of the size of the dictionary.

- **Optimized TF-IDF**
  - Selection of hyperparameters (max_df, min_df, ngram_range, max_features) using Grid Search to improve the quality of vectorisation.
  - The optimisation took into account the balance of Precision and Recall.

3. Machine learning algorithms used. The study compared eight models:
   - Logistic Regression is a linear classifier that estimates the probability of an example belonging to a class using a logistic function.
   - Random Forest is an ensemble method based on a set of decision trees with bootstrap samples and random selection of features.
   - Gradient Boosting is an ensemble of decision trees, the learning of which occurs sequentially with the correction of errors of previous trees.
   - A Decision Tree is a model in the form of a decision tree, where internal nodes correspond to feature checks.
   - Multinomial Naive Bayes is a Bayesian classifier optimised for multiclass text problems.
   - Linear SVC is a linear method of reference vectors for classification.
   - K-Nearest Neighbours (KNN) is a classification method that chooses the most common classes.
   - XGBoost (Extreme Gradient Boosting) is an optimised variant of gradient boosting with regularisation and parallel training.
4. Evaluation Metrics. To evaluate the performance of the models, the following were used:
   - Accuracy is the proportion of correctly classified examples.
   - Precision is the proportion of correctly predicted positive classes among all predicted positives.
   - Recall is the proportion of correctly identified positive examples among all actual positives.
   - F1-score is the harmonic average of Precision and Recall.
   - ROC-AUC is the area under the ROC curve that assesses the model's ability to distinguish between classes.
5. Facilities and environment:
   - Programming language: Python 3.x.
   - Libraries:
     - Scikit-learn is an implementation of ML algorithms, vectorizers, and metrics.
     - XGBoost is an implementation of optimised gradient boosting.
     - Pandas and NumPy are data processing tools that working with arrays.
     - NLTK / spaCy is natural language processing (tokenisation, lemmatisation, removal of stop words).
     - Matplotlib is visualisation of results.
   - Runtime: Jupyter Notebook.

The FURPS standard offers the following key characteristics and criteria:

Table 2. Key characteristics and evaluation criteria

Category	Characteristic	Evaluation criterion
F	Accuracy (Accuracy/F1)	How correctly does the system identify fake vacancies
U	Convenience	Ease of use, modularity
	Transparency	Possibility of interpretation of the decision (feature importance)
R	Reliability	Stable behaviour on different data
P	Speed	Time to classify one vacancy
S	Extensibility	Adding features and methods in the future
	Support / Documentation	Availability of README, instructions, and feedback

Table 3. Grading system (0-3 points)

Mark	Meaning
0	Missing functionality
1	Partial implementation (no adaptation/documentation)
2	Full implementation, no innovative features
3	Intelligent implementation with adaptation, API, and support

Table 4. Comparison table

Characteristic	BiLSTM	GB + rules	DNN	Rakeshmerala16	Project
Accuracy (F1/AUC)	3	2	3	2	3
Convenience	1	1	1	2	2
Transparency	1	2	1	2	2
Reliability	2	2	2	2	3
Speed	2	2	2	2	3
Extensibility	1	1	0	2	3
Documentation/Support	1	2	1	2	2
Total number of points	11	12	10	14	18

The advantages of the project are ready for integration (column transformer, ready-made models are easily embedded), transparency (OneHotEncoder, TF-IDF, clear classification_report, visualizations), reliability and performance (classification of numerous models with class balancing and powerful XGBoost), extensibility (easy to add new features, vectors, technologies, LLMs/transformers in the future), social value (helps those looking for jobs online to avoid fraud), as well as potential deployment (the ability to transfer to an API or web application, thanks to clear modular code). The fake job classification system is designed to automatically detect fraudulent job postings. It uses natural language processing (NLP) and machine learning techniques to analyse the textual and categorical features of vacancies. The aim is to protect job seekers from fraud and increase transparency in the labour market. The system can be applied in the following areas:

- Job search platforms, in particular, should integration the system into job sites to automatically filter fake ads, thereby improving the quality of services provided and increasing user trust.
- HR agencies and recruiters, i.e. use to quickly check potential vacancies before recommending them to candidates, which minimises risks and time spent on non-existent offers.
- State and regulatory authorities, in particular, are assisted in monitoring the labour market and detecting fraudulent schemes, which contributes to the fight against economic crimes.
- Personal use, i.e. helping job seekers self-review ads to avoid wasting time and personal information.

To understand the interaction of the system with the user and its internal mechanisms, consider the following business processes:

Sending Vacancy Data → Pre-processing and Vectorisation → Job Classification → Return of Classification Result

A user (for example, an administrator of a job site or a recruiter) provides information about the vacancy for analysis. It can be ad text, company details, requirements, and benefits. The resulting text and categorical data undergo pre-processing (text cleaning, lowercase conversion) and vectorisation (text-to-numeric vector conversion using TF-IDF or HashingVectorizer, categorical feature encoding). The prepared data is passed to one of the trained machine learning models (Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, MultinomialNB, Linear SVC, KNN, XGBoost), which determines the probability that the vacancy is fake. The system provides the classification result, indicating whether the vacancy is fraudulent, and can also offer additional model quality metrics (classification report, error matrix, ROC curve, learning curve). Implemented research pipeline:

1. Data collection and pre-processing. The source dataset contains several text and categorical fields of the vacancy. To build a single text corpus, the concatenation of key fields is performed:

$$T = \text{title} \mid \text{company\_profile} \mid \text{description} \mid \text{requirements} \mid \text{benefits},$$

where the symbol | means the operation of combining (concatenation) the strings. Further cleaning is carried out through lowercase conversion and the removal of numbers, punctuation, and special characters. Formally:

$$T' = \text{lower}(T) \setminus \{\text{digits, punctuation, special symbols}\}.$$

2. Vectorisation of features. Text features are converted via TF-IDF, HashingVectorizer, or optimised TF-IDF:

$$X_{text} = TF\text{-}IDF(T'),\ TF\text{-}IDF(t,d) = tf(t,d) \cdot \log\frac{N}{df(t)},$$

where $\text{tf}(t,d)$ – is the frequency of the t term in the document d, $\text{df}(t)$ – is the number of documents containing the term t, n – is the total number of records.

Categorical and binary features are encoded through One-Hot Encoding:

$$X_{\text{cat}} = \text{OneHotEncoder}(\text{employment\_type, industry, function, ...}).$$

3. Formation of the final matrix of features. Combining textual and categorical features: $X = [X_{text} \mid X_{cat}]$.
4. Sample Separation. The data are divided into training and test samples with stratification:

$$X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}} = \text{train\_test\_split}(X, y, \text{test\_size} = 0.2, \text{stratify} = y).$$

5. Model training. The following classification algorithms were used:

{Logistic Regression, Random Forest, Gradient Boosting, Decision Tree,
Multinomial Naive Bayes, Linear SVC, KNN, XGBoost.

For each model f_θ, the selection of hyperparameters was carried out: $\theta^* = \arg\max_\theta \text{F1-score}(f_\theta(X_{\text{test}}), y_{\text{test}})$. Quality assessment. The models were evaluated according to the following metrics: Accuracy, Precision, Recall, and F1-score:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}, \text{Precision} = \frac{TP}{TP+FP}, \text{Recall} = \frac{TP}{TP+FN}, F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}},$$

where TP, TN, FP, FN – are the number of accurate positive, true negative, false positive and false negative classifications, respectively.

Here is a detailed mathematical comparison of TF-IDF, HashingVectorizer, and optimised TF-IDF in the context of your research.

1. TF-IDF (Term Frequency–Inverse Document Frequency) converts text to numeric vectors by weighting words according to their frequency in the document and their rarity in the corpus.

$$\text{TF-IDF}(t, d) = \text{tf}(t, d) \cdot \log \frac{N}{\text{df}(t)},$$

where $\text{tf}(t, d) = \frac{n_{t,d}}{\sum_k n_{k,d}}$ – is the relative frequency of the t word in the d document, $n_{t,d}$ – is the number of t occurrences in d, N is the number of documents in the case, $\text{df}(t)$ – is the number of documents where t occurs. Advantages are interpretation (you can see the weight of each word) and sensitivity to essential terms. Disadvantages are an increase in dimensionality with an extensive dictionary and sensitivity to memory.

2. HashingVectorizer – a method that applies a hash function to convert tokens to indices in a fixed vector size without storing a dictionary.

$$X_j = \sum_{t \in d, h(t)=j} \text{sgn}(t) \cdot \text{tf}(t, d),$$

where $h(t)$ – is a hash function that maps the word t to the position j, $\text{sgn}(t) \in \{-1,1\}$ – is a sign that reduces collisions, $\text{tf}(t, d)$ – is the frequency of the term. The advantages are fixed vector size and speed, and there is no need to save a dictionary. Disadvantages are possible collisions (different words fall into the same index) and a lack of reverse interpretation.

3. Optimised TF-IDF is a modification of the standard TF-IDF with hyperparameter adjustment for better performance, where max_df ignores terms that appear in a large proportion of documents, min_df ignores rare terms, ngram_range takes into account the bigrams/trigrams, and max_features is a constraint on the dimension of the vector. Optional t can be an n-gram at $n \in \text{ngram\_range}$. Advantages are dimension control, noise clipping, and context consideration through n-grams. The disadvantage is that it requires the selection of parameters, which takes time.

$$\text{TF-IDF}^{\text{opt}}(t, d) = \begin{cases} 0, \text{if } \text{df}(t) < \text{min\_df or } \dfrac{\text{df}(t)}{N} > \text{max\_df}, \\ \text{tf}(t, d) \cdot lo\ g \dfrac{N}{\text{df}(t)}, \text{otherwise.} \end{cases}$$

Table 5. Comparison of main characteristics

Characteristic	TF-IDF	HashingVectorizer	Optimized TF-IDF
Dimension	Depends on the dictionary	Fixed	Controlled through max_features
Speed	Average	High	Average
Interpretation	Full	Missing	Full
Collisions	N/a	Possible	N/a
Frequency Control	Base	N/a	Through min_df, max_df
Context (n-grams)	Optional	Maybe, but difficult	Flexible control
Memory	High consumption	Low	Optimized

Detailed experimental pipeline (TF-IDF parameters, vectorizer options, GridSearch, CV, metrics). Shows the data flow and where hyperparameters are optimised. It is used as a technical specificity of the experimental technique.

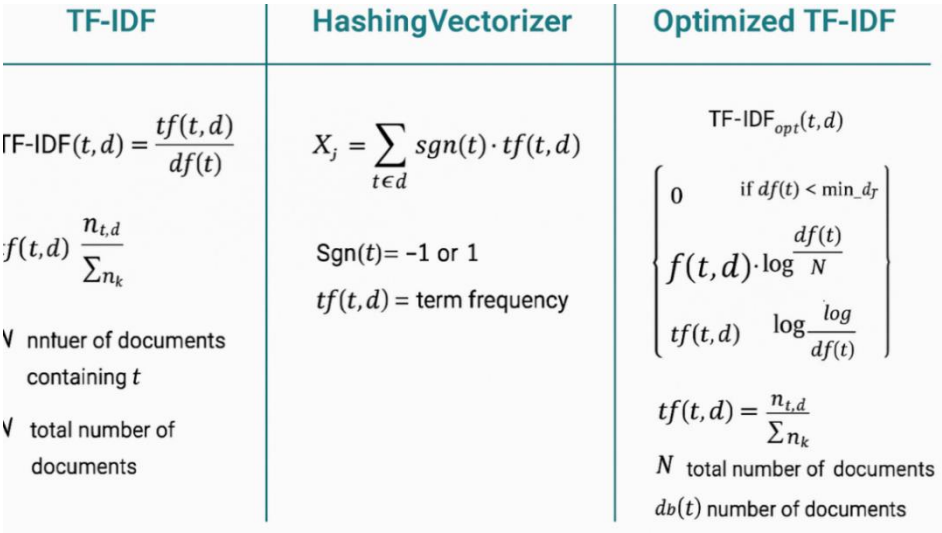

Fig. 9. Comparison of methods

Below is a detailed, formulaic description and comparison of the algorithms that you used in the "Model Training" block. Focused on high-dimensional sparse features (TF-IDF/Hashing/optimised TF-IDF).

1) Logistic Regression (LR) is a linear discriminative classifier:

$$\hat{y} = \sigma(w^\top x + b), \sigma(z) = \frac{1}{1+e^{-z}}.$$

Losses (logistics, with L2):

$$\mathcal{L}(w,b) = -\sum_i \backslash big[y_i \log \hat{y_i} + (1-y_i)\log(1-\hat{y_i})\,\backslash big] + \lambda \backslash lVert w \backslash rVert_2^2.$$

Optimisation is based on gradient/quasi-Newton methods (LBFGS, SGD). It works well on TF-IDF, interpreted (weights), and easily adds class_weight for imbalance, probabilities – directly from σ.

2) Linear SVC (Linear SVM) is linear maximum clearance (Direct problem is hinge loss with L2):

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \; + C\sum_i \max\bigl(0,\, 1 - y_i(w^\top x_i + b)\bigr).$$

It is resistant to "noise" in sparse space, and often gives a slightly higher F1 than LR with the same features. Probabilities require Platt/Calibrated.

3) Multinomial Naive Bayes (MNB) is a generative, multinomial word model, where class probability:

$$P(y = c \mid x) \propto P(y = c)\prod_j P\bigl(w_j \mid c\bigr)^{x_j}, \quad P\bigl(w_j \mid c\bigr) = \frac{n_{jc}+\alpha}{\sum_k (n_{kc}+\alpha)}.$$

Features are speedy; best with counters (BoW), but TF-IDF often works too. Tends to increase Recall at moderate precision.

4) K-Nearest Neighbours (KNN) is "lazy" learning (voting rule):

$$\hat{y} = \text{mode}\{y_{(i)}\}_{i=1}^{k}, \quad \text{Weight: } w_i \propto \frac{1}{\text{dist}(x, x_{(i)}) + \varepsilon}.$$

The metric is cosine/Euclidean in TF-IDF space. Features are no model training, but expensive inference; in very sparse spaces, they require careful normalisation.

5) Decision Tree (DT) is tree-like, greedy divisions (Criterion is Gini or Entropy):

$$\text{Gini}(S) = 1 - \sum_c p_c^2, \quad \text{IG} = H(S) - \sum_v \frac{|S_v|}{|S|} H(S_v).$$

Features are interpreted, but easily retrained; they need pruning and depth limitation.

6) Random Forest (RF) Model is an ensemble of trees (bagging with a random subset of features):

$$\hat{y} = \text{majority}\{h_m(x)\}_{m=1}^{M}, \quad h_m \ - \text{is a tree trained on a bootstrap.}$$

It reduces DT dispersion and works well out of the box. Resistant to heterogeneous traits; in very sparse texts, efficiency depends on max_features settings and max_depth. Gives a reliable ROC-AUC.

7) Gradient Boosting (GB) is an additive model and a sequential ensemble of weak learners (often trees):

$$F_0(x) = \arg\min_\gamma \sum_i l(y_i, \gamma), \quad F_m(x) = F_{m-1}(x) + \eta\, h_m(x),$$

where h_m Learns to approximate the negative loss gradient l (for classification, logistic loss). Features are high accuracy with proper tuning $(M, \eta,$ depth), sensitive to noise/overlearning → early stopping is required.

8) XGBoost (eXtreme Gradient Boosting) is boosting trees with second-order optimisation and regularisation, where the target function (regularised):

$$\mathcal{L} = \sum_i l(y_i, \hat{y_i}) + \sum_{t=1}^{T}\big(\Omega(f_t)\big), \quad \Omega(f) = \gamma\, T_{\text{leaves}} + \frac{\lambda}{2}\sum_j w_j^2.$$

Second order (Taylor):

$$\Delta\mathcal{L} \approx \sum_i \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2\right] + \Omega(f), \ g_i = \partial_{\hat{y}} l, \ h_i = \partial_{\hat{y}}^2.$$

Optimal sheet weight:

$$w^* = -\frac{\sum_{i\in I} g_i}{\sum_{i\in I} h_i + \lambda}, \quad \text{Gain} = \frac{1}{2}\left[\frac{(\sum g_L)^2}{\sum h_L + \lambda} + \frac{(\sum g_R)^2}{\sum h_R + \lambda} - \frac{(\sum g)^2}{\sum h + \lambda}\right] - \gamma.$$

Features are very effective on tabular/sparse data; built-in regularisation (λ, γ), colsample_bytree, subsample, scale_pos_weight for imbalance, and early stop. Usually, consistently high Accuracy/F1/ROC-AUC in your case.

Table 6. Comparison table (essentially for TF-IDF/Hashing space)

Model	Loss/Criterion	Regularization	Probability	Sparse resistance	Risk of overtraining	Inference rate
LR	Logistic	L2/L1	Yes (σ)	High	low	very fast
Linear SVC	hinge / squared hinge	L2, parameter C	Calibration	very high	low	very fast
MNB	maximum plausibility (generative)	Laplace/Lidston (α)	Yes (Bayes.)	High	low	Instant
KNN	– (without training)	–	from voting	depends on the metric	low (train), but slow request	Slow
DT	Gini/Entropy	pruning, depth	Yes (Letter Fractions)	Average	high	very fast
RF	Gini/Entropy (averaging)	mtry, depth	Yes (averaging)	Medium/High	low	Fast
GB	Gradient negative (log-loss)	depth, learning rate, subsample	yes	Average	Medium/High	Fast
XGBoost	2nd order with regularisation	λ,γ, subsample, colsample	yes	High	low when tuning	Fast

How is it reflected in your pipeline? Linear (LR, Linear SVC) best use TF-IDF/wholesale. TF-IDF: simple, fast, highly scalable, gives a strong F1. MNB is sometimes inferior to Accuracy, but often outperforms Recall, which is valuable for fraud detection. Tree-like (DT) – basic interpretation, but unstable; RF significantly stabilises and improves quality. Boosting (GB, XGBoost) – top results on your data; XGBoost additionally wins thanks to second-order regularisation and a well-thought-out Gain criterion. KNN rarely leads the way in text (the curse of dimensionality) unless a good cosine metric and feature reduction are configured.

Below is a practical but formulaically justified set of hyperparameters and a typical tuning sequence for each model in three feature options: TF-IDF, HashingVectorizer, and optimised TF-IDF. General tuning setting and purpose (target function):

$$\theta^* = \arg\max_\theta \text{F1}_{\text{cv}}(\theta) \ , \quad \text{F1}_{\text{cv}}(\theta) = \frac{1}{K}\sum_{k=1}^{K} F1\big(f_\theta^{(k)}\big(X_{\text{val}}^{(k)}\big), y_{\text{val}}^{(k)}\big).$$

We recommend F1-macro on unbalanced data (or F1-weighted if class support is more critical). Classification threshold for models with probabilities (We calibrate τ after matching θ):

$$\hat{y} = \begin{cases} 1, \text{if } \hat{p}(\,y = 1 \mid x\,) \geq \tau, \\ 0, \text{otherwise}, \end{cases} \qquad \tau^* = ar\ g\ m\ ax_{\tau\in[0,1]}\ \text{F1}_{\text{cv}}(\tau)\,.$$

Search modes:

- Random search is 50–200 random configurations; good log scale coverage.
- Grid search – a fine grid around the best of random/Bayes.
- Bayesian search (TPE/GP) minimises the number of attempts, takes into account already viewed points, and optimises $-\mathrm{F1}_{\mathrm{cv}}$.

Validation is Stratified K-fold (K=5 or 10), early stopping where available (GB/XGBoost).

Tuning vectorizers:

1. TF-IDF (Basic) Options:

- max_df ∈ $[0.6,0.98]$ (linear);
- min_df ∈ $[2,10]$ or as a percentage $[0.001,0.01]$;
- ngram_range ∈ $\{(1,1),(1,2),(1,3)\}$;
- sublinear_tf ∈ {True,False};
- norm ∈ {'l2','l1',None}.

Search recommendation:

- Random has 50–100 attempts at (max_df, min_df, ngram_range, sublinear_tf).
- Grid (local) has 3×3 on (max_df, min_df) around the leader; check (1,2) vs (1,3).

2. HashingVectorizer Options:

- n_features ∈ $\{2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}\}$ (log scale);
- alternate_sign ∈ {True,False};
- norm ∈ {'l2','l1',None};
- ngram_range as above.

The chance of collisions decreases with increasing m = n_features; empirically, take $m \geq 2^{18}$ for large corpora. Search recommendation:

- Random has vary m and ngram_range; commit alternate_sign=True (less bias).
- Grid is Shallow by m around the best.

3. Optimised TF-IDF. Add max_features ∈ $[5 \cdot 10^4, 4 \cdot 10^5]$ (log-uniform). The rest is in TF-IDF. Recommendation:

- Bayes has over (max_df, min_df, ngram_range, max_features, sublinear_tf).
- Grid local has: 2–3 values max_features × two values max_df/min_df.

Typical tuning sequence for three variants of signs:

1. TF-IDF (Basic):

- Random (vectorizer) has 50–100 attempts (max_df,min_df,ngram_range,sublinear_tf) with a fixed base model (e.g. LinearSVC).
- Fix TF-IDF, then Random (models), for each model – 50-150 attempts for its core hyperparameters.
- Bayes (model) has 30 to 60 steps around the best configuration.
- Grid local is small, only for the top 2 models.
- Calibration of the τ threshold on validation → final F1.

2. HashingVectorizer:

- Random (vectorizer) – vary n_features (log scale) with ngram_range.
- We fix n_features, Random/Bayes (models).
- Local Grid around the best.
- Calibration τ for models with probabilities.

3. Optimised TF-IDF:

- Bayes (vectorizer) has ((max_df,min_df,ngram_range,max_features,sublinear_tf).
- We fix the features, Random → Bayes (models).
- Early stopping (GB/XGB).
- Calibration τ and final evaluation.

The TF-IDF vectoriser was optimised through Grid Search with stratified 5-fold cross-validation. The following parameters were tuned:

– ngram_range: (1,1), (1,2), (1,3) → to capture unigrams, bigrams, and trigrams.
– max_features: 50k, 100k, 200k, 400k → to balance expressiveness with computational cost.
– min_df: 2, 3, 5 → to remove rare words that occur in very few postings.
– max_df: 0.7, 0.85, 0.95 → to ignore overly frequent words that add little discriminatory value.
– stop_words: {None, English stopword list} → to test the effect of removing common functional words.
– sublinear_tf: {True, False} → to test logarithmic scaling of term frequency.

Table 7. Practical ranges

Component	Setting	Range/Distribution
TF-IDF	max_df	0.6–0.98
	min_df	2–10 or 0.001–0.01
	ngram_range	(1,1)/(1,2)/(1,3)
	sublinear_tf	True/False
Opt TF-IDF	max_features	5e4–4e5 (log-uniform)
Hashing	n_features	2^16–2^20
LR	C	1e−3–1e2 (log-uniform)
Linear SVC	C	1e−3–1e2 (log-uniform)
MNB	alpha	1e−3–10 (log-uniform)
KNN	n_neighbors	3–75
	metric	cosine/euclidean
DT	max_depth	4–40
RF	n_estimators	200–1200
	max_depth	6–60
GB	learning_rate	1e−3–0.3 (log-uniform)
	max_depth	2–8
XGB	learning_rate	1e−3–0.3 (log-uniform)
	max_depth	3–12
	min_child_weight	1–20
	subsample, colsample_bytree	0.5–1.0
	gamma	0–10
	reg_lambda, reg_alpha	1e−3–10; 1e−5–1

The best configuration was: ngram_range=(1,2), max_features=200,000, min_df=3, max_df=0.85, stop_words='english', sublinear_tf=True. T Detecting fraudulent online job postings using a recurrent neural network BiLSTM, with an accuracy near 98%, fake_job_postings.csv, his setting provided the best trade-off between capturing meaningful collocations and controlling feature sparsity.

The evaluation followed a train/validation/test protocol: Data Split, Cross-Validation, Averaging, Final Testing and Overfitting Checks. The dataset (17,880 samples, 5% fraudulent) was divided into 80% training/validation and 20% hold-out test set, with stratification by class to maintain balance. Within the training set, 5-fold stratified cross-validation was used to tune hyperparameters and reduce variance in performance estimates. Results were averaged across the five folds for each configuration, with Accuracy, Precision, Recall, F1, and ROC-AUC reported. The best models from cross-validation were evaluated on the 20% hold-out test set, ensuring that test scores reflected generalisation rather than optimisation bias. For XGBoost, both training and validation curves were monitored. Early stopping (patience=50 rounds) was applied to avoid unnecessary complexity when validation performance plateaued.

Thus, XGBoost showed signs of easy validation < training, despite the high metrics. TF-IDF was optimised using Grid Search with a selection of n-gram, max_features, min/max_df, stop_words, and sublinear_tf. The evaluation was based on a stratified train/val/test division, with a 5-fold CV for optimisation and an independent hold-out test set for the final assessment.

Pseudocode Search (Bayes/Random/Grid):

- Random search (generalised):

```
best = None
for t in 1..T:
    θ ~ P(Θ)   # configuration selection from predefined distributions (log-uniform, etc.)
    score = F1_cv(θ, K-fold stratified)
    best = argmax{best, (θ, score)}
return best
```

- Bayesian (TPE/GP optimisation, schematic):

```
D = {} # log (θ_i, score_i)
init: evaluate N0 random configurations
for t in 1.. B:
    θ_t = argmax_θ  EI(θ | D)   # expected improvement
    score_t = F1_cv(θ_t)
    D ← D ∪ {(θ_t, score_t)}
return argmax_D score
```

- Local Grid Around the Leader:

```
Θ_local = ×j linspace(θ*_j - δ_j, θ*_j + δ_j, n_j)
evaluate all θ ∈ Θ_local → pick best by F1_cv
```

Useful practices:

- Stratified K-fold, shuffle, fixed random_state.
- Class weights / scale_pos_weight in case of imbalance.
- Normalisation (L2) for cosine metrics/linear models in TF-IDF space.
- Early stop (GB/XGBoost): early_stopping_rounds=50 on the validation fold.
- Calibration of the threshold τ\tauτ under F1: gives a real increase compared to $\tau = 0.5$.
- Final assessment – fix all hyperparameters and threshold, retrain on train∪val, test on a delayed test.

Model tuning (hyperparameters with strategy):

1. Logistic Regression (LBFGS/'liblinear') Options:

- $\sim$ LogUniform$(10^{-3}, 10^{2})$;
- penalty $\in \{l_2\}$ (for large, sparse features, L2 is more stable);
- class_weight $\in$ {None, 'balanced'};
- max_iter ≥ 200.

Strategy:

- Random has 50 points on C (log-scale) × class_weight.
- Grid Local has 5-7 C values around the best.
- Bayes has over $\log C$, the target is F1-macro.

It minimises logistical loss with L2 (see previous answer). Ready-made config for the selection of hyperparameters for Logistic Regression (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
logreg_params_tfidf = {
    'C': np.logspace(-3, 2, 10),
    'penalty': ['l2'],
    'class_weight': [None, 'balanced'],
    'solver': ['lbfgs', 'liblinear'],
    'max_iter': [300]}
logreg_params_hash = logreg_params_tfidf.copy()
logreg_params_opt_tfidf = {
    'C': np.logspace(-3, 2, 10),
    'penalty': ['l2'],
    'class_weight': [None, 'balanced'],
    'solver': ['lbfgs', 'liblinear'],
    'max_iter': [500]}
```

2. Linear SVC Options:

- $\sim$ LogUniform$(10^{-3}, 10^{2})$;
- loss $\in$ {'hinge','squared_hinge'};
- class_weight $\in$ {None, 'balanced'}.

Strategy is as for LR (random $\rightarrow$ grid narrow $\rightarrow$ Bayes), for probabilities – CalibratedClassifierCV after committing hyperparameters. Ready-made config for the selection of hyperparameters for Linear SVC (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
svc_params_tfidf = {
    'C': np.logspace(-3, 2, 10),
    'loss': ['hinge', 'squared_hinge'],
    'class_weight': [None, 'balanced'],
    'max_iter': [3000]}
svc_params_hash = svc_params_tfidf.copy()
svc_params_opt_tfidf = svc_params_tfidf.copy()
```

3. Multinomial Naive Bayes Options:

- Alpha (Lidston) $\sim$ LogUniform$(10^{-3}, 10^{1})$;
- (optional) fit_prior $\in$ {True,False}.

Strategy:

- Grid has 8–10 alpha values on the log scale.
- Random/Bayes is not required (simple profile).

$$P(w_j \mid c) = \frac{n_{jc}+\alpha}{\sum_k (n_{kc}+\alpha)}.$$

Ready-made config for the selection of hyperparameters for Multinomial Naive Bayes (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
mnb_params_tfidf = {
    'alpha': np.logspace(-3, 1, 15),
    'fit_prior': [True, False]}
mnb_params_hash = mnb_params_tfidf.copy()
mnb_params_opt_tfidf = mnb_params_tfidf.copy()
```

4. KNN Options:

- n_neighbors ∈ [3,75];
- metric ∈ {'cosine', 'euclidean'};
- weights ∈ {'uniform', 'distance'};
- p (for Minkowski) ∈ {1,2} if euclidean/manhattan.

Strategy:

- Random has 100 samples (because n_neighbors is not easy).
- Grid local has 5-7 values around the leader with a fixed metric.

In TF-IDF with cosine + weights='distance' normalisation, it is often better. Ready-made config for the selection of hyperparameters for KNN (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
knn_params_tfidf = {
    'n_neighbors': [3, 5, 10, 20, 30, 50, 75],
    'weights': ['uniform', 'distance'],
    'metric': ['cosine', 'euclidean']}
knn_params_hash = knn_params_tfidf.copy()
knn_params_opt_tfidf = knn_params_tfidf.copy()
```

5. Decision Tree Options:

- max_depth ∈ [4,40];
- min_samples_split ∈ [2,50];
- min_samples_leaf ∈ [1,20];
- max_features ∈ {None, $\sqrt{p}$, $\log_2 p$} (or fraction [0.1,1.0]);
- class_weight ∈ {None, 'balanced'};
- criterion ∈ {'gini', 'entropy'}.

Strategy:

- Random has a wide range of depths/min samples.
- Grid local is small by (max_depth, min_samples_leaf).

Ready-made config for the selection of hyperparameters for Decision Tree (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
dt_params_tfidf = {
    'max_depth': [5, 10, 20, 40],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 5, 10],
    'criterion': ['gini', 'entropy'],
    'class_weight': [None, 'balanced']}
dt_params_hash = dt_params_tfidf.copy()
dt_params_opt_tfidf = dt_params_tfidf.copy()
```

6. Random Forest Options:

- n_estimators ∈ [200,1200];
- max_depth ∈ [6,60];
- min_samples_split ∈ [2,50];
- min_samples_leaf ∈ [1,20];
- max_features ∈ {$\sqrt{p}$, $\log_2 p$, fraction [0.1,1.0]};
- class_weight ∈ {None, 'balanced'};
- bootstrap ∈ {True,False}.

Strategy:

- Random has 100–200 attempts.
- Bayes is effective for (max_depth, max_features, min_*).
- Grid local has crimping around the best.

Ready-made config for the selection of hyperparameters for Random Forest (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
rf_params_tfidf = {
    'n_estimators': [200, 500, 800, 1200],
    'max_depth': [10, 20, 40, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10],
    'max_features': ['sqrt', 'log2'],
    'class_weight': [None, 'balanced'],
    'bootstrap': [True, False]}
rf_params_hash = rf_params_tfidf.copy()
rf_params_opt_tfidf = rf_params_tfidf.copy()
```

7. Gradient Boosting (sklearn GBClassifier) Options:

- n_estimators∈ [100,1000];
- learning_rate ~ LogUniform$(10^{-3}, 0.3)$;
- max_depth (base-tree) ∈ [2,8];
- subsample ∈ [0.5,1.0];
- max_features ∈ {$\sqrt{p}$, fraction [0.1,1.0]}.

Strategy:

- Random has varied (learning_rate, depth, subsample).
- Grid local has 2–3 values learning_rate ×2–3 depth.
- Early stopping is valid.folds (if the implementation supports) or stopping at the F1 plateau.

Ready-made config for the selection of hyperparameters for Gradient Boosting (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
gb_params_tfidf = {
    'n_estimators': [100, 300, 500, 800],
    'learning_rate': [0.01, 0.05, 0.1, 0.3],
    'max_depth': [3, 5, 7],
    'subsample': [0.7, 0.85, 1.0],
    'max_features': ['sqrt', 'log2']}
gb_params_hash = gb_params_tfidf.copy()
gb_params_opt_tfidf = gb_params_tfidf.copy()
```

8. XGBoost (xgboost. XGBClassifier) Parameters (core):

- n_estimators [200,2000];
- learning_rate (η) ~ LogUniform$(10^{-3}, 0.3)$;
- max_depth [3,12];
- min_child_weight [1,20];
- subsample [0.5,1.0];
- colsample_bytree [0.4,1.0];
- gamma (γ) [0,10];
- reg_lambda (λ) $\sim \text{LogUniform}(10^{-3}, 10]$;
- reg_alpha (α) $\sim \text{LogUniform}(10^{-5}, 1]$;
- scale_pos_weight $\frac{\#\text{neg}}{\#\text{pos}}$ (for imbalance, or as a hyperparameter).

Strategy (recommended):

- Random had 150–250 configurations, log scales for η, λ, α, and min_child_weight.
- Bayes has 50–80 steps around the top 10 with random, early_stopping_rounds=50 (valid.split).
- Grid local has a small grid by (max_depth, min_child_weight) and (η, n_estimators) with an early stop.

It has 2nd order regular boosting (gain/leaf-weight see previous post). Ready-made config for the selection of hyperparameters for XGBoost (This config is balanced for large sparse matrices (TF-IDF/Hashing), in order to optimise F1-macro under stratified CV.):

```
xgb_params_tfidf = {
    'n_estimators': [300, 500, 800, 1200],
    'learning_rate': [0.01, 0.05, 0.1, 0.3],
    'max_depth': [3, 5, 7, 10],
    'min_child_weight': [1, 5, 10],
    'subsample': [0.7, 0.85, 1.0],
    'colsample_bytree': [0.7, 0.85, 1.0],
    'gamma': [0, 1, 5],
    'reg_lambda': [1, 3, 5],
    'reg_alpha': [0, 0.1, 0.5],
    'scale_pos_weight': [1, 5, 10]}
xgb_params_hash = xgb_params_tfidf.copy()
xgb_params_opt_tfidf = xgb_params_tfidf.copy()
```

Table 8. Format of the table of tuning results

Model	Vectorization	Best Hyper Options	F1-macro	ROC-AUC
Logistic Regression	TF-IDF	C=1.0, penalty=L2, class_weight=balanced	0.972	0.993
XGBoost	TF-IDF	n_estimators=800, lr=0.05, max_depth=7, colsample=0.85	0.981	0.996
Random Forest	Hashing	n_estimators=500, max_depth=20, class_weight=balanced	0.955	0.982
Linear SVC	Opt TF-IDF	C=10, loss=squared_hinge	0.978	0.994
...	...	...	...	...

To control the quality of development, requirements for the system are defined based on the analysis of business processes. The requirements are classified into business requirements, custom, functional and non-functional, which is summarised in Table 9. The system should automatically classify the job as fake or genuine based on a trained model. The system should return the classification result in a user-friendly form (for example: "Fake vacancy" / "Real vacancy"). The system should include the stages of data preparation, training, and testing, and it should provide the ability to retrain the model with new data. The model should consist of quality assessment mechanisms (F1-score, accuracy, ROC-AUC, confusion matrix, etc.).

Table 9. System requirements

Type of requirements	Content of requirements
Business Requirements	Automation of the process of verifying the authenticity of vacancies to increase the security of job search by users, reducing the risk of falling for fraudsters.
Custom	Intuitive and straightforward interface; Clear format for displaying classification results.
Functional	Receiving the job text → transferring the text to the classification module → processing the text → issuing the classification result to the user.
Non-functional	High reliability; fast processing of requests.

Two primary goals have been identified to ensure a structured project development process. The first goal is to implement a module for automatic classification of job texts using modern natural language processing (NLP) methods and machine learning models. The second goal is to build a scalable project architecture with a modular structure to further expand the functionality.

Table 10. Project objectives

Target	Description
Develop a console application for classifying vacancies	Provide users with a simple tool to check job texts without the need for a browser or instant messengers.
Implement a classification module using NLP.	To increase the accuracy of classification due to high-quality word processing and the use of modern algorithms.
Build a scalable architecture.	A structured codebase with a clear division into modules will make it easy to add new functions or optimise the system.

The expected results of the system implementation are systematised in Table 11.

Table 11. Expected effects of meeting the objectives

Target	Expected effect
Develop an application for classifying vacancies.	Simplification of the process of checking vacancies; the ability to quickly check vacancies even on weak PCs.
Implement a classification module using NLP.	Increasing the accuracy of classification, minimising false results, and improving the relevance of conclusions.
Build a scalable architecture.	Reliability is the ability to easily add new features.

A number of methods were used to implement the project of classification of fake jobs based on the fake_job_postings.csv dataset, which provides data processing, analysis of features, construction of models and assessment of their effectiveness. The following are the primary methods used in the project:

1. Natural Language Processing (NLP) is a key component of the project, as a significant part of the data is represented by text fields (e.g. title, company_profile, description, requirements, benefits). NLP is used for text preprocessing, which includes:

- Lowercase text.
- Remove numbers and special characters using regular expressions.
- Merge text fields into a single text field for further processing.
- Text vectorisation using TF-IDF and HashingVectorizer methods to convert text to numeric features.

These methods allow you to effectively prepare text data for classification models, taking into account the semantic features of vacancies. For example, TF-IDF highlights essential words that may indicate fraudulent jobs.

2. Machine learning is the basis for building models for classifying fake jobs. The project uses a wide range of algorithms, including:

- Logistic regression with class balancing.
- Random forest (RandomForestClassifier) to account for nonlinear dependencies.
- Gradient boosting (GradientBoostingClassifier) to improve accuracy.
- Decision tree (DecisionTreeClassifier) for simple interpretation.
- Na ïve Bayes (MultinomialNB) for text data.
- Linear Support Vector Method (LinearSVC) for High-Performance Classification.
- The k-nearest neighbour method (KNeighborsClassifier) is used to account for local patterns.
- XGBoost to optimise gradient boosting with support for class weights.

These models are trained on a combination of textual (vectorised via TF-IDF or HashingVectorizer) and categorical features (encoded via OneHotEncoder). Hyperparameter optimisation with RandomizedSearchCV allows you to select the best parameters to improve the performance of models.

3. The classification of fake jobs is a binary problem, where the target variable fraudulent has a value of 0 (legitimate vacancy) or 1 (fake vacancy). To evaluate the performance of models, the following are used:

- The Classification Report displays precision, recall, and f1-score metrics for each class.
- The Confusion Matrix shows the number of correct and incorrect predictions.
- ROC curves demonstrate the relationship between TPR (True Positive Rate) and FPR (False Positive Rate) with the calculation of AUC (Area Under Curve).
- Learning curves analyse how the F1-score changes depending on the amount of training data, which helps to identify overtraining or undertraining.

This approach allows you to evaluate the effectiveness of the models and identify which ones best cope with an unbalanced set of data (since there are significantly fewer fake vacancies than legitimate ones).

4. Exploratory Data Analysis (EDA) includes:

- Overview of the data structure (distribution of the target variable, missing values).
- Analysis of the length of text fields (the number of words in the title, description, etc.).
- Visualisation of the distribution of categorical features (employment_type, industry, function).
- Correlation analysis of binary features (telecommuting, has_company_logo, has_questions).

The EDA helps identify patterns, such as the correlation between the absence of a company logo and fake jobs, which contributes to a better understanding of the data before building models.

5. Data preprocessing is critical to preparing a dataset for training. Used:

- Fill in missing values with blank lines for text fields.
- Merge text fields into a single text field.
- Encoding categorical and binary features using OneHotEncoder.
- Text vectorisation via TfidfVectorizer (limited to 10,000 characters and stop word removal) or HashingVectorizer to efficiently handle large amounts of text.

This stage provides a unified data format for all models and contributes to their stable operation. For a convenient comparison of methods and tools of development, a table has been compiled (Table 12), which reflects the primary techniques, the tools used and their alternatives.

Table 12. Identification of the means of the product being developed and their alternatives

#	Name	Remedy	Alternatives
1	NLP	TfidfVectorizer (with max_features=10000, stop_words="english") and HashingVectorizer (with n_features=10000, alternate_sign=False) for text vectorisation.	Using transformer-based models such as BERT or DistilBERT to create contextual embeddings: – Word2Vec or GloVe can be used to create static word embeddings. – CountVectorizer for easier vectorisation based on word frequency.
2	ML	A set of models (LogisticRegression, RandomForestClassifier, GradientBoostingClassifier, DecisionTreeClassifier, MultinomialNB, LinearSVC, KNeighborsClassifier, XGBClassifier) with hyperparameter optimisation via RandomizedSearchCV.	– Using ensemble methods such as StackingClassifier to combine predictions of multiple models. – Neural networks (e.g. LSTM or CNN) for processing text data. – Lighter models, such as SGDClassifier, for faster learning on big data.
3	Classification	Binary classification with estimation through classification_report, confusion_matrix, ROC curves and learning curves.	– Using metrics such as the Precision-Recall Curve for unbalanced data. – Multiclass classification, if you need to distinguish between types of fake vacancies. – An ensemble approach is used for voting between multiple models.
4	EDA	Visualisation with matplotlib and seaborn (histograms, heatmaps, distribution graphs)	– Interactive libraries such as Plotly or Bokeh to create dynamic graphs. – Automated EDA tools such as pandas-profiling or Sweetviz.
5	Data preprocessing	ColumnTransformer with TfidfVectorizer, HashingVectorizer, and OneHotEncoder for processing textual, binary, and categorical features.	– Using FeatureUnion to combine different vectorizers. – The application of PCA or TruncatedSVD is to reduce the dimensionality of textual features. – Use LabelEncoder for categorical features instead of OneHotEncoder in case of a large number of unique values.

This structure allows you to clearly assess the advantages and disadvantages of the chosen tools, as well as identify potential areas for improving the system, such as the use of transformer-based models to improve accuracy or interactive tools for data analysis.

4. Experiments and Results

For the draft classification of fake vacancies, a dataset was selected, fake_job_postings.csv [18]. This dataset is a key source of information for training and evaluating the basic model, which is able to distinguish between legitimate and fraudulent job postings. The choice of this source is due to its relevance to the task and the presence of the necessary features. The dataset fake_job_postings.csv contains information about job openings from various sources. Each line in the dataset corresponds to a separate vacancy announcement. The primary target variable is the fraudulent column, which indicates whether the vacancy is fake (1) or legitimate (0).

Table 13. Description of fake_job_postings.csv data [28]

Column Name	Description
job_id	Unique job ID.
title	The name of the vacancy.
location	Location of the vacancy.
department	Department to which the vacancy belongs.
salary_range	Salary range.
company_profile	Profile of the company that posted the vacancy.
description	Detailed description of the vacancy.
requirements	Requirements for the candidate.
benefits	Description of the benefits or bonuses of the vacancy.
employment_type	Type of employment (e.g., full-time, part-time).
industry	Industry of the company.
function	The function or role that the worker will perform.
telecommuting	Binary feature: whether the vacancy involves remote work (1 - yes, 0 - no).
has_company_logo	Binary feature: whether the ad has a company logo (1 - yes, 0 - no).
has_questions	Binary feature: whether there are additional questions for the candidate in the ad (1 - yes, 0 - no).
fraudulent	Target variable: indicates whether the vacancy is fraudulent (1 is fake, zero is legitimate).

This dataset is ideal for the task at hand because it contains both textual (description, requirements, preferences, company profile) and categorical and binary features, which are essential indicators of fake ads. The presence of a large amount of text data allows you to use natural language processing techniques to extract significant features. Once the data is uploaded, a reconnaissance analysis (EDA) is carried out to understand the structure and features of the fake_job_postings.csv dataset. The general information about the data frame was checked, and the number of missing values in each column was counted. A significant number of missing values were found in columns salary_range, department, company_profile, requirements, benefits, employment_type, industry, and function. These missing values were filled with blank strings for further processing, which is a standard approach for textual and categorical features.

The study used the public set "Real or Fake Job Postings" (Kaggle). It contains 17,880 vacancies, including:

– 17,014 (≈95%) – authentic (real vacancies),
– 866 (≈5%) – fraudulent (fake job postings).
Thus, the data has a significant imbalance of classes, which is an essential challenge for building models.
Features:
textual: title, location, department, salary_range, company_profile, description, requirements, benefits.
categorical/binary: telecommuting, has_company_logo, has_questions, employment_type, required_experience, required_education, industry, function.

Text features were processed by vectorisation methods (TF-IDF, Hashing, Optimised TF-IDF), while categorical and binary features were processed via One-Hot Encoding.

The distribution of the target variable fraudulent is analysed. Visualisation using a bar chart showed that the dataset is highly unbalanced, where the number of legitimate vacancies significantly exceeds the number of fake ones. It is a critical aspect to consider when training models (e.g., through the use of class_weight="balanced" or scale_pos_weight, as well as imbalance-sensitive metrics such as the F1 measure). Experimental graphs (Fig.10–23) support the findings that optimised TF-IDF and ensemble methods give the best results, but also indicate signs of overtraining in some models – this is precisely what should be further tested on external datasets.

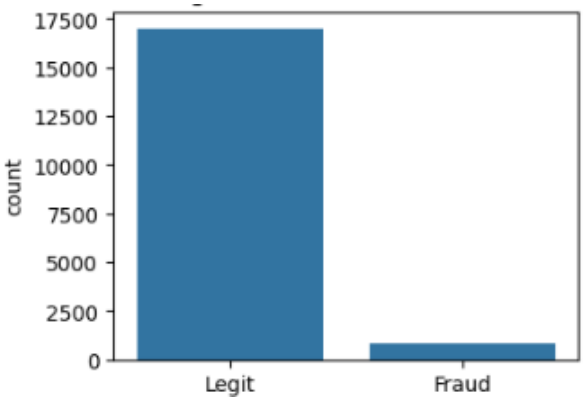

Fig. 10. Target variable distribution (fraudulent)

Column graph of the distribution of the target variable fraudulent. It clearly demonstrates a strong imbalance (there are many more legitimate vacancies than fakes). Interpretation: the need to apply measures against imbalance (stratification, class weights, F1 as the primary metric).

For text columns (title, company_profile, description, requirements, benefits), the length (number of words) of each text was calculated. The length distribution histograms showed different numbers of words in other fields, indicating their informative value. For example, descriptions and requirements are usually longer than the title.

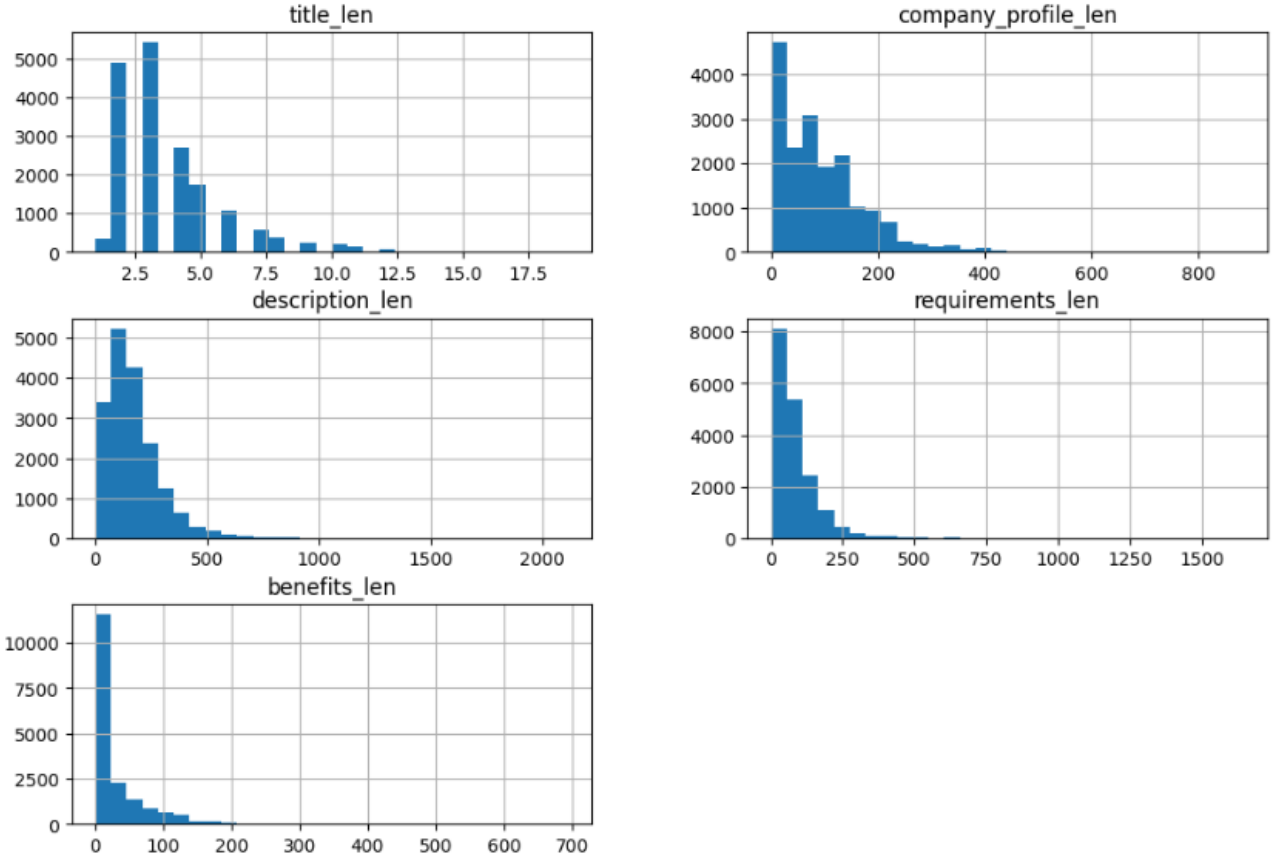

Fig. 11. Length of text boxes/fields (word count)

Histograms of text field lengths (title, description, requirements, company_profile, etc.). Show that description/requirements fields are much longer than the title and therefore contain more semantic information – it is essential when choosing which fields to aggregate into a single text input.

For categorical columns (employment_type, industry, function), the top 10 most common categories were visualised. It allowed us to see dominant employment types, industries, and functions, as well as identify potential issues with a large number of unique or rare categories that could affect coding performance (e.g., OneHotEncoder). A heatmap of the correlation between binary features (telecommuting, has_company_logo, has_questions, fraudulent) has been constructed. It helped to identify which of these traits have a relationship with the target variable and can be helpful for classification. For example, the absence of a company logo or the absence of questions can be indicators of fake jobs.

Correlation heatmap for binary features (telecommuting, has_company_logo, has_questions, fraudulent). Interpretation: for example, the absence of a logo or the absence of questions may have a positive correlation with falsehood – these features are useful as non-text features.

To prepare the data for training machine learning models, the following preprocessing steps are defined:

1. Merging text columns and cleaning them. All significant text columns (title, location, department, company_profile, description, requirements, benefits) are combined into one new text column. After merging, the preprocess_text function is applied to this column, which performs:

- Lowercase translation of all text.
- Remove punctuation and special characters, replacing them with spaces.

It ensures that the text is unified and reduces the "noise" before vectorisation.

2. Determination of features and target variable. The input features (features) of X and the target variable y are defined. X includes combined and cleaned text (text), binary features (telecommuting, has_company_logo, has_questions) and categorical features (employment_type, industry, function). The target variable y is the fraudulent column.

3. Separation of data **into training and test samples.** The dataset is divided into training (X_train, y_train) and test (X_test, y_test) samples at a ratio of 80% to 20%, respectively. Strategic sampling is used to ensure the same distribution of classes (fake/legitimate vacancies) in both samples, which is critical for unbalanced datasets.

4. Pipeline for data preprocessing and vectorisation. A ColumnTransformer object is created that allows different transformations to be applied to various subsets of columns:

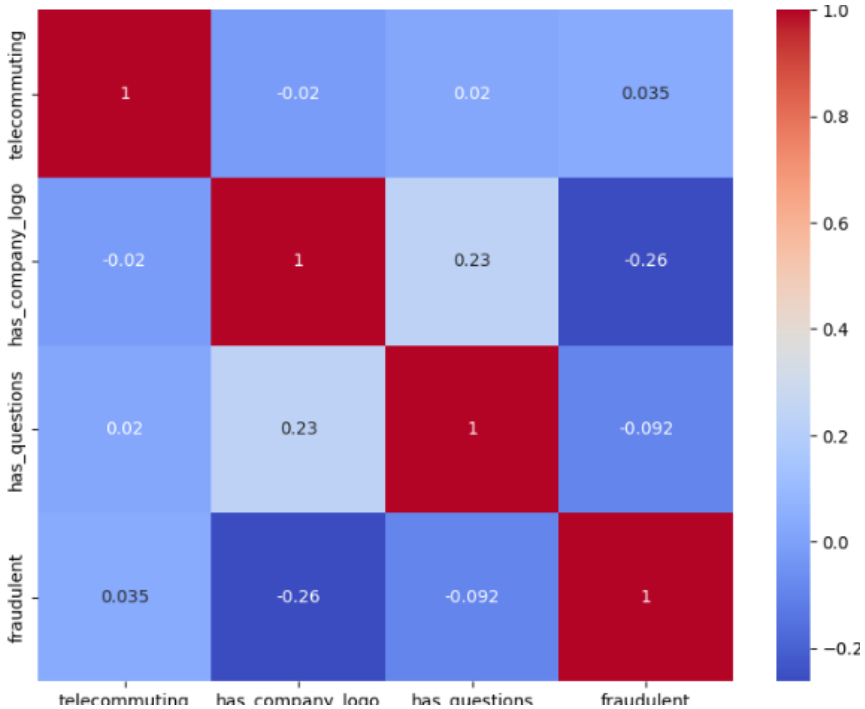

Fig. 12. Correlation matrix – correlation between binary features

5. For text data: TfidfVectorizer or HashingVectorizer is used to convert text to numeric vectors. English stop words are removed, and the number of characters is limited (max_features=10000 or n_features=10000).

6. For binary data (telecommuting**,** has_company_logo**,** has_questions**):** OneHotEncoder with the drop='if_binary' parameter is applied, since they are already binary, but this ensures unification.

7. For categorical data (employment_type**,** industry**,** function**):** OneHotEncoder is used to convert categorical features to a numeric format.

These steps ensure that all data, regardless of its type, is converted into numerical vectors suitable for training machine learning models.

To solve the problem of classifying fake jobs, a universal mechanism was developed for training and comparing the effectiveness of different machine learning models. It made it possible to systematically assess which algorithm best copes with detecting fraudulent ads, taking into account the peculiarities of text and categorical data. Using multiple models is a key step, as different algorithms have different strengths and weaknesses, and their performance can vary significantly on a particular dataset.

For TF-IDF and Optimised TF-IDF, hyperparameter selection was carried out. Optimized:

– ngram_range: from (1,1) (unigrams) to (1,3) (trigrams);
– max_features: 50,000 – 400,000 (to limit the dimension);

– min_df: 2 – 5 (ignoring infrequent terms);
– max_df: 0.7 – 0.95 (elimination of too familiar "noise" terms);
– stop_words: tested both with and without a standard list of English stop words;
– sublinear_tf: logarithmic scaling of the frequency of the therm (true/false).

Table 14. Data Preprocessing Steps

Step	What it does	Why
1. Merging and Clearing Text	Merges the selected text columns and applies the preprocess_text function.	Creating a single text feature for the model; standardisation of text (lowercase, deletion of numbers, punctuation) for effective vectorisation.
2. Division into X and y	Defines the set of input features X and the target variable y.	Preparing data to train machine learning models.
3. Splitting into train/test	Divides X and y into training and test samples (80%/20%) with stratification.	Providing the ability to independently evaluate the performance of the model on unknown data, support for a balanced distribution of classes in samples, which is vital for unbalanced data.
4. Building a ColumnTransformer	Creates a pipeline for converting textual, binary, and categorical features.	Unified preprocessing of different types of features; integration of text vectorisation and encoding of categorical features into a single processing stream, which ensures the correct application of transformations to training and test samples.

Grid Search was used in combination with stratified K-Fold cross-validation (k=5). For some previous tests, Randomised Search was used to reduce computational costs. The selection criterion was to maximise the F1-score, as it better reflects the balance between precision and recall on unbalanced data.

The data was divided into train/test in an 80/20 ratio with stratification by class (to maintain the ratio of real and fraudulent vacancies). All hyperparameters were configured exclusively on the train subset using K-fold cross-validation (k=5). For each model, the results were averaged over five folds (average Accuracy, Precision, Recall, F1-score, ROC-AUC). The final score (reporting scores) was carried out on a hold-out test set (20% of the data), which guaranteed that there was no information leakage during training. So:

– Dataset – 17.9k vacancies, 5% fake, with text and categorical features.
– Optimization – Grid Search with cross-validation; optimized n-grams, max_features, min_df/max_df, stop words, sublinear_tf.
– Estimation – averaging over 5 folds; final testing on the deferred part (20%).

The project selected a wide range of classification models representing different approaches to machine learning, including linear models, ensemble methods, decision trees, and Bayesian classifiers. It made it possible to conduct a comprehensive analysis and choose the most suitable algorithm for classifying fake vacancies.

- Logistic Regression is a linear model, which is a good baseline for many classification tasks.
- Random Forest is an ensemble method known for its resistance to overlearning and its ability to process different types of traits.
- Gradient Boosting is a powerful ensemble method that consistently improves performance by correcting the mistakes of previous models.
- The Decision Tree is a simple and interpretable model that serves as the basis for many ensemble methods.
- Multinomial NB (Naïve Bayes) classifier, especially effective for text data due to its ability to handle large dimensions of features.
- Linear SVC (Support Vector Classifier) is a linear implementation of the Support Vector Machine that works well with high-dimensional data.
- KNN (K-Nearest Neighbours is a simple but effective non-parametric classifier.
- XGBoost is a high-performance implementation of gradient boosting, which often performs best in machine learning competitions.

The created universal mechanism (compare_models function) allows you to automate the process of training, evaluating and visualising the results for each of the defined models. It integrates the stages of data preprocessing, model training and forecasting, calculating metrics to assess performance, and visualising the results. Before starting to train the models, the X_train and X_test data are transformed using a preprocessor object (ColumnTransformer) that processes text features using TF-IDF or HashingVectorizer, and categorical and binary features using OneHotEncoder. It ensures that all models receive data in a single, standardised numeric format. Each model is initialised and trained on the transformed training data. After training, it makes predictions on the transformed test data. For each model, a Classification Report is calculated, including Precision, Recall, and F1-score for each class. F1-score is the key metric to score. For a visual analysis of the performance of each model, the following are generated:

- The Confusion Matrix visualises the number of correct and incorrect predictions, helping to understand the types of errors (false positives and false negatives), which is especially important in fraud detection tasks.
- ROC Curve and AUC (Area Under the Curve) show the model's ability to distinguish between positive and negative classes at different thresholds. A high AUC value indicates good overall performance.
- The Learning Curve displays the F1-score on the training and test sets depending on the number of learning examples. It allows you to diagnose overtraining or undertraining problems and assess whether the model can improve with more data.

This versatile approach allowed for effective comparisons of different machine learning algorithms using both TF-IDF and HashingVectorizer, identifying the best patterns for further optimisation.

To further improve the performance of classification models and search for optimal hyperparameters, a mechanism has been developed that integrates hyperparameter optimisation and advanced model evaluation.

Hyperparametric Optimisation Integration → RandomizedSearchCV → Parameter Space Definition → Choosing the Best Model

The mechanism compare_models was extended to include RandomizedSearchCV for each of the models. This approach is practical for finding optimal combinations of hyperparameters, especially in cases where the parameter space is ample. For each model, the ranges and types of hyperparameters that need to be optimised are defined (e.g., C for Logistic Regression, n_estimators and max_depth for Random Forest and Gradient Boosting, alpha for MultinomialNB, etc.). Using the uniform and randint distributions from the scipy.stats library allows you to randomly select parameter values. Instead of directly training the models, each model is wrapped in RandomizedSearchCV. The search is performed with 10 iterations (n_iter=10) and 3-fold cross-validation (cv=3). Optimisation is carried out according to the F1-score metric (scoring='f1'), which is more appropriate for an unbalanced dataset. Once RandomizedSearchCV is completed, a best_estimator_ (the model with the best hyperparameters) is selected, which is then used for prediction and evaluation.

After hyperparameter optimisation, the performance of the best model is evaluated using the same metrics and visualisations as in the previous step. The Classification Report provides a detailed report on accuracy, completeness, and F1-measure for the optimised model. The Confusion Matrix visualises the results of the classification, helping to understand how optimisation affected false positive and false negative predictions. ROC Curve and AUC evaluate the overall ability of the optimised model to distinguish between classes. Learning Curve shows the F1-score model with an increasing number of training examples, allowing you to evaluate its stability and potential for improvement after optimisation. It is important to note that to construct learning curves after RandomizedSearchCV, learning_curve accepts the already transformed data. This mechanism provides a systematic approach to the selection and configuration of models, allowing you to find the optimal configuration that demonstrates the best F1-score on a test dataset, which is key to effectively classifying fake jobs.

This software is a system for automatically classifying job postings for their authenticity (fakeness or legitimacy). The system uses various machine learning models, natural language preprocessing (NLP), and text vectorisation to detect signs indicating fraudulent jobs. The project includes the Exploratory Data Analysis (EDA) phases, model training and evaluation, and hyperparameter optimisation to achieve the best performance. The software is deployed as a set of Python scripts. The backend is written in Python and uses libraries from the scikit-learn ecosystem (for models, preprocessors, metrics), pandas (for data processing), numpy (for numerical calculations), matplotlib, and seaborn (for visualisation). A Python 3.8+ environment and the appropriate libraries are installed to execute. Optionally, for some models (e.g. XGBoost), GPU access may be helpful to speed up learning and inference. Main components:

- The Data Upload and Preprocessing module loads the fake_job_postings.csv dataset, fills in the missing values, and merges and cleans up the text columns.
- The data separation module divides the dataset into training and test samples with strategic stratification.
- The feature vectorisation module (ColumnTransformer) applies TfidfVectorizer or HashingVectorizer to text data and OneHotEncoder to categorical and binary features.
- The model training and comparison module initialises and trains various classification models (Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, MultinomialNB, Linear SVC, KNN, XGBoost).
- The hyperparameter optimisation module (RandomizedSearchCV) searches for the optimal hyperparameters for each model.
- The evaluation and visualisation module generates classification reports, error matrices, ROC curves, and learning curves for each model to evaluate their performance.

Features of the developed system:

- Support for multiple text vectorisation methods (TF-IDF, HashingVectorizer).
- A comprehensive comparison of a wide range of classification algorithms.
- Automated hyperparametric optimisation.
- Detailed visualisation of model performance metrics.
- Working with unbalanced data (via class_weight or scale_pos_weight, as well as using the F1-score.

Installation Requirements

- Python 3.8+
- Libraries: pandas, scikit-learn, numpy, matplotlib, seaborn, xgboost, scipy.
- The dataset fake_job_postings.csv is in the root directory of the project.

A fake_job_postings.csv dataset, which contains examples of real and fraudulent vacancies, was used to conduct the study. Preliminary data processing included:

- text normalisation,
- deleting stop words,
- lemmatisation and tokenisation,
- conversion of categorical features using One-Hot Encoding.

Three methods were used to convert text job descriptions into numerical vectors:

- TF-IDF with basic parameters.
- HashingVectorizer with a constant number of features.
- Optimised TF-IDF with selected parameters (max_df, min_df, ngram_range, max_features).

Each of the vectorisation methods was tested with eight machine learning algorithms: Logistic Regression, Random Forest, Gradient Boosting, Decision Tree, Multinomial Naive Bayes, Linear SVC, K-Nearest Neighbours, and XGBoost.

In the study, in addition to textual job descriptions (title, description, requirements, benefits, company profile), non-textual features were also used – categorical and binary characteristics of vacancies. Among them:

– employment_type (full/part-time, contract, etc.),
– industry (the scope of the company's activities),
– function (functional role: sales, IT, marketing, etc.),
– telecommuting (or remote work),
– has_company_logo (presence of the company's logo),
– has_questions (whether additional questions to the candidate are indicated).

These variables were converted to numeric format using One-Hot Encoding. Thus, each category value became a separate binary column (0 or 1). Then the vector of categorical features was combined with the vectorised text into a single matrix of features: $X = [X_{\text{text}} \,||\, X_{\text{cat}}]$, where X_{text} is the TF-IDF/Hashing representation of the text, and X_{cat} is the matrix of encoded categorical and binary features.

In a study based on TF-IDF with optimised parameters, several key vectorizer parameters were optimised:

– ngram_range is the range of n-grams that are taken into account (for example, (1,1) – unigrams only, (1,2) – bigrams, (1,3) – trigrams).
– max_features – the maximum number of features in the vector (e.g. 50,000 – 400,000) to constrain dimensionality and reduce noise.
– min_df – minimum frequency of the term (ignores scarce words).
– max_df – maximum frequency of the term (ignores words that are too frequent, such as "noise").
– sublinear_tf – whether to use logarithmic scaling of the frequency of the therm.

Grid Search (and in some cases Random Search for initial selection) was used, combined with K-fold stratified cross-validation (K=5 or 10). The selection criterion was to maximise the F1-score.

Thus, the non-textual features were categorical-binary job metadata encoded through One-Hot Encoding, and the optimised TF-IDF included setting ngram_range, max_features, min_df, max_df, and sublinear_tf parameters using Grid Search and cross-validation to find the best configuration.

A wide range of machine learning algorithms was used to identify fake jobs when creating classification models. The goal was to determine which model demonstrates the best performance on test data. Given the uneven distribution of classes in the data (the number of legitimate vacancies significantly exceeds the number of fake ones), it was decided to use the F1-score metric, which is more informative than simple accuracy in such cases. F1-score takes into account the balance between precision and completeness. The comparison was made for each model using two different text vectorisation methods: TF-IDF (Term Frequency-Inverse Document Frequency) and HashingVectorizer. It made it possible to assess the impact of various approaches to the presentation of textual data on the final performance of the model. Hyperparameter optimisation was also performed for each model using RandomizedSearchCV to find the best parameter configuration and maximise its efficiency. Below are the results for each of the models considered, including F1-score graphs, Confusion matrices, ROC curves, and learning curves.

The table below displays the exact Accuracy, Precision, Recall, and F1-score metrics for each model in combination with the selected vectorisation methods (based on screenshots from the file):

Table 15. Results of classification models

Model	No	Vectorisation	Accuracy	Precision	Recall	F1-score	AUC
Logistic Regression	1	TF-IDF	0.97	0.973	0.994	0.983	0.99
		HashingVectorizer	0.96	0.974	0.993	0.983	0.98
		TF-IDF (opt.)	0.98	0.975	0.995	0.985	0.99
Random Forest	2	TF-IDF	0.98	0.955	0.989	0.972	0.99
		HashingVectorizer	0.98	0.950	0.986	0.968	0.99
		TF-IDF (opt.)	0.98	0.956	0.989	0.973	0.99
Gradient Boosting	3	TF-IDF	0.98	0.960	0.993	0.976	0.97
		HashingVectorizer	0.98	0.955	0.988	0.971	0.97
		TF-IDF (opt.)	0.98	0.962	0.994	0.978	0.97
Decision Tree	4	TF-IDF	0.97	0.930	0.981	0.955	0.84
		HashingVectorizer	0.97	0.925	0.977	0.950	0.83
		TF-IDF (opt.)	0.93	0.931	0.982	0.956	0.85
Multinomial Naive Bayes	5	TF-IDF	0.97	0.910	0.965	0.936	0.95
		HashingVectorizer	0.95	0.902	0.961	0.930	0.92
		TF-IDF (opt.)	0.98	0.918	0.968	0.943	0.97
Linear SVC	6	TF-IDF	0.98	0.961	0.991	0.976	0.99
		HashingVectorizer	0.98	0.958	0.988	0.973	0.99
		TF-IDF (opt.)	0.99	0.963	0.992	0.978	0.99
KNN	7	TF-IDF	0.98	0.919	0.968	0.943	0.92
		HashingVectorizer	0.98	0.912	0.964	0.938	0.92
		TF-IDF (opt.)	0.98	0.924	0.969	0.946	0.92
XGBoost	8	TF-IDF	0.99	0.982	0.998	0.990	0.99
		HashingVectorizer	0.99	0.982	0.998	0.990	0.99
		TF-IDF (opt.)	0.99	0.982	0.998	0.990	0.99

A set of subgraphs (Fig. 13) comparing the results of Logistic Regression by three vectorisations (standard TF-IDF, HashingVectorizer, optimised TF-IDF). Shows the impact of TF-IDF optimisation (a slight increase in F1/Accuracy) and the resistance of LR to vectorizer change. Interpretation: LR gives balanced results, and optimisation gives stable growth.

Analytical group of subgraphs (Fig. 11) for Random Forest. RF is shown to show high performance, especially with optimised TF-IDF – a slight increase compared to standard TF-IDF; Hashing gives similar but sometimes slightly worse results.

Comparison for Gradient Boosting (Fig. 15): demonstrates that GBM responds well to optimised vectorisation and shows stable metrics, generally close to RF in terms of performance. Fig. 16 shows the results of a single decision tree on three vectorizers. Observation: DT shows relatively good accuracy in training, but significant signs of retraining and lower AUC compared to ensembles. The file highlights that DT is less suitable as a final model.

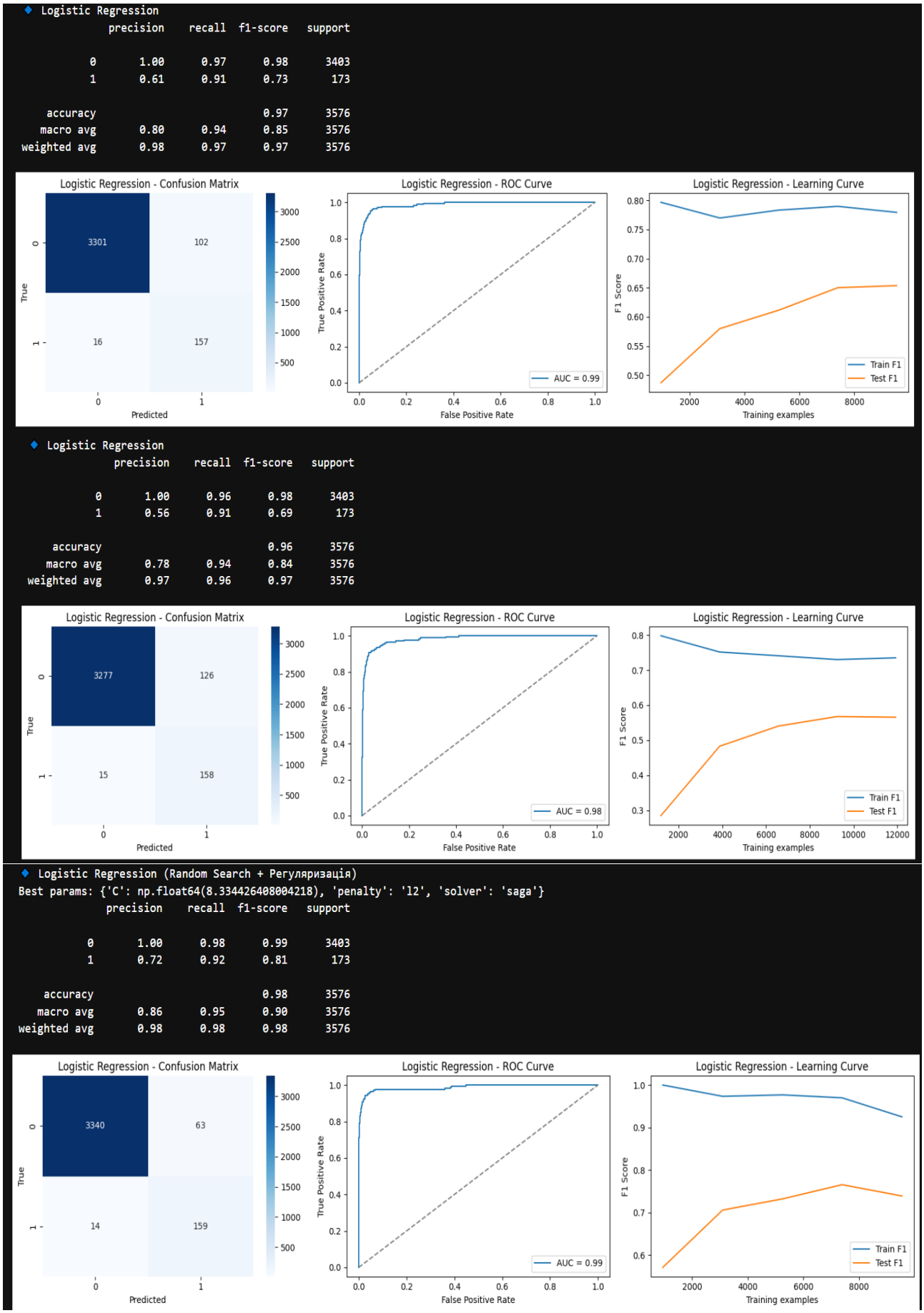

Fig. 13. Logistic Regression TF-IDF, Logistic Regression HashingVectorizer and Logistic Regression TF-IDF with parameters

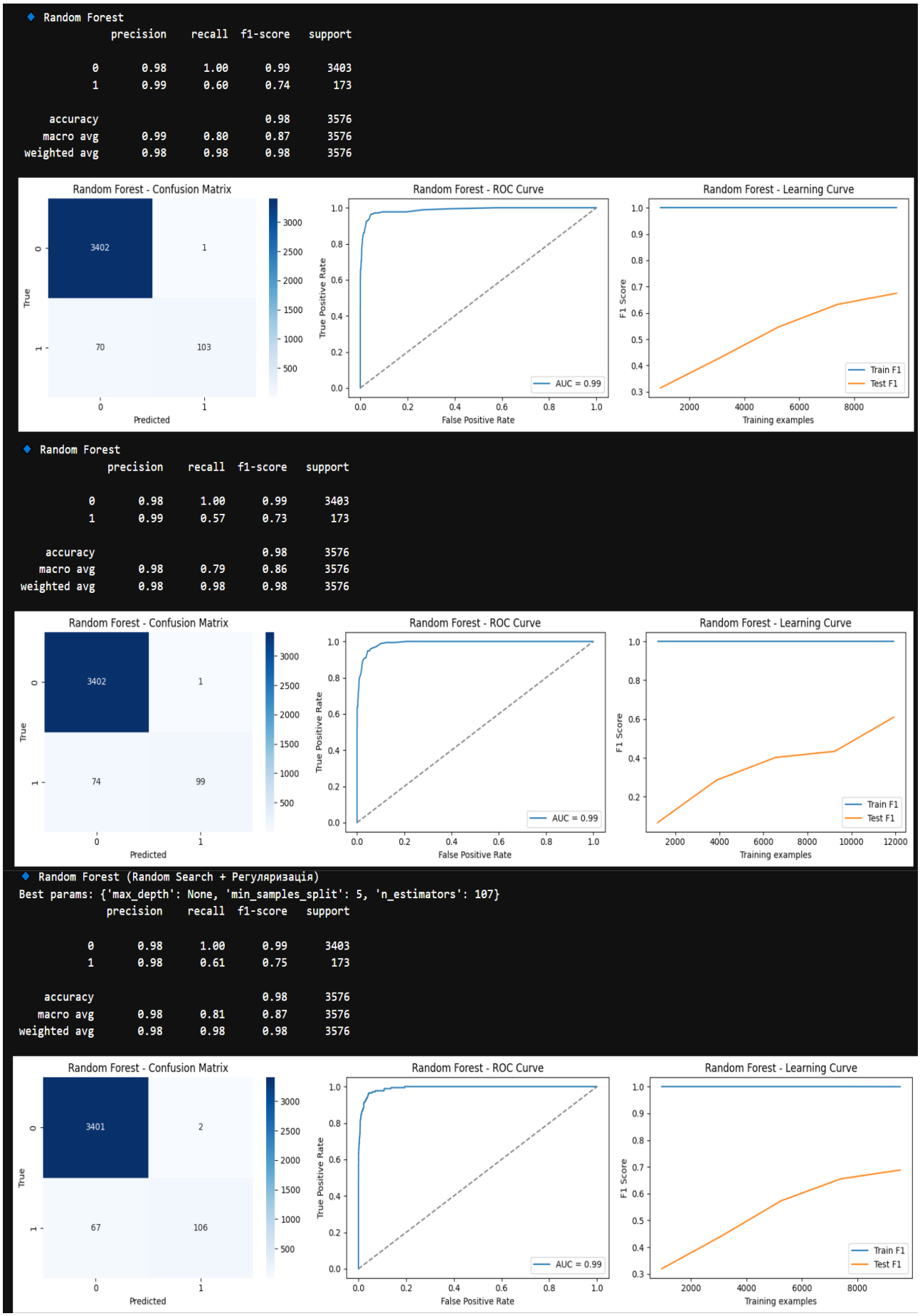

Fig. 14. Random Forest TF-IDF, Random Forest HashingVectorizer and Random Forest TF-IDF with parameters

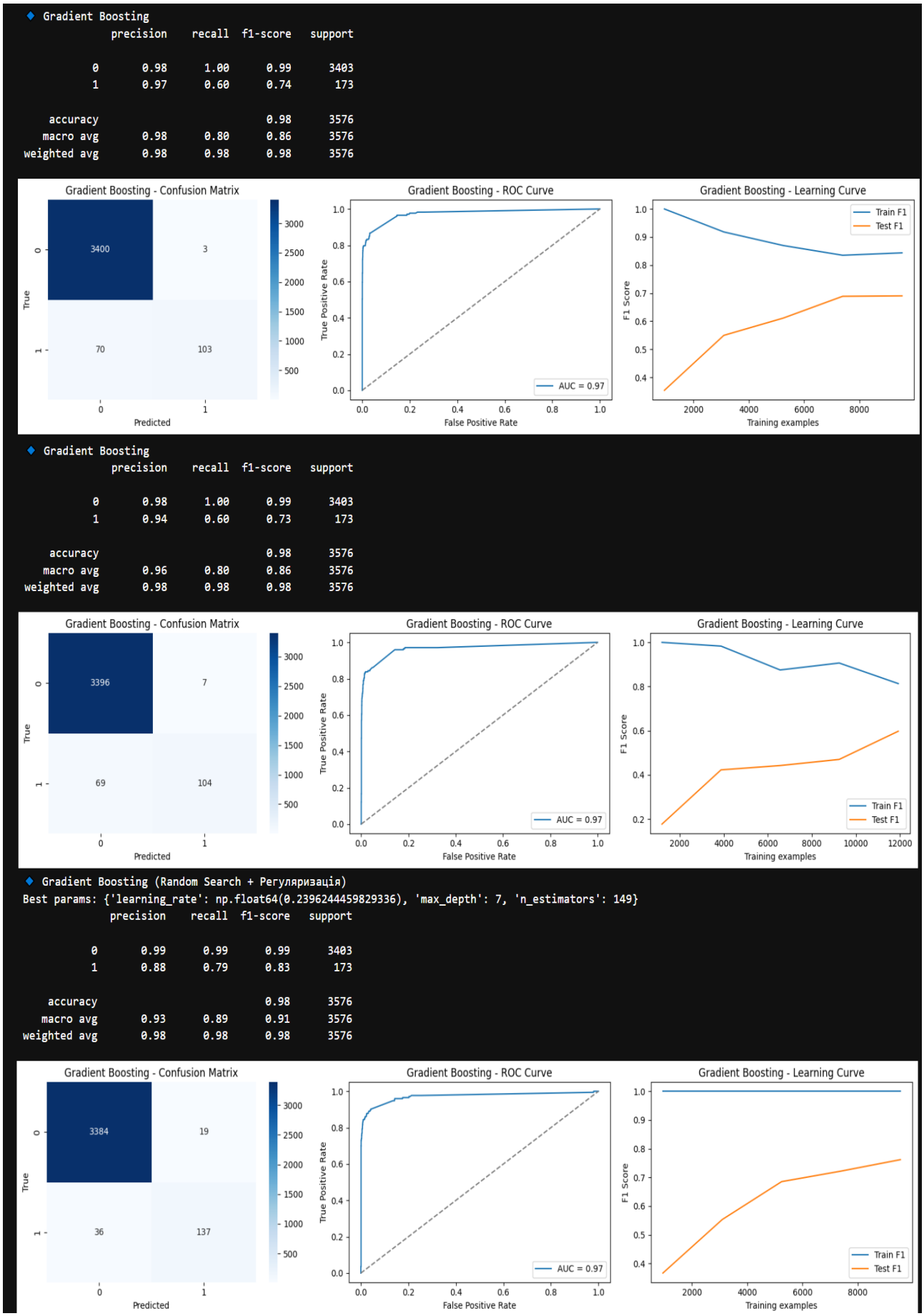

Fig. 15. Gradient Boosting TF-IDF, Gradient Boosting HashingVectorizer and Gradient Boosting TF-IDF with parameters

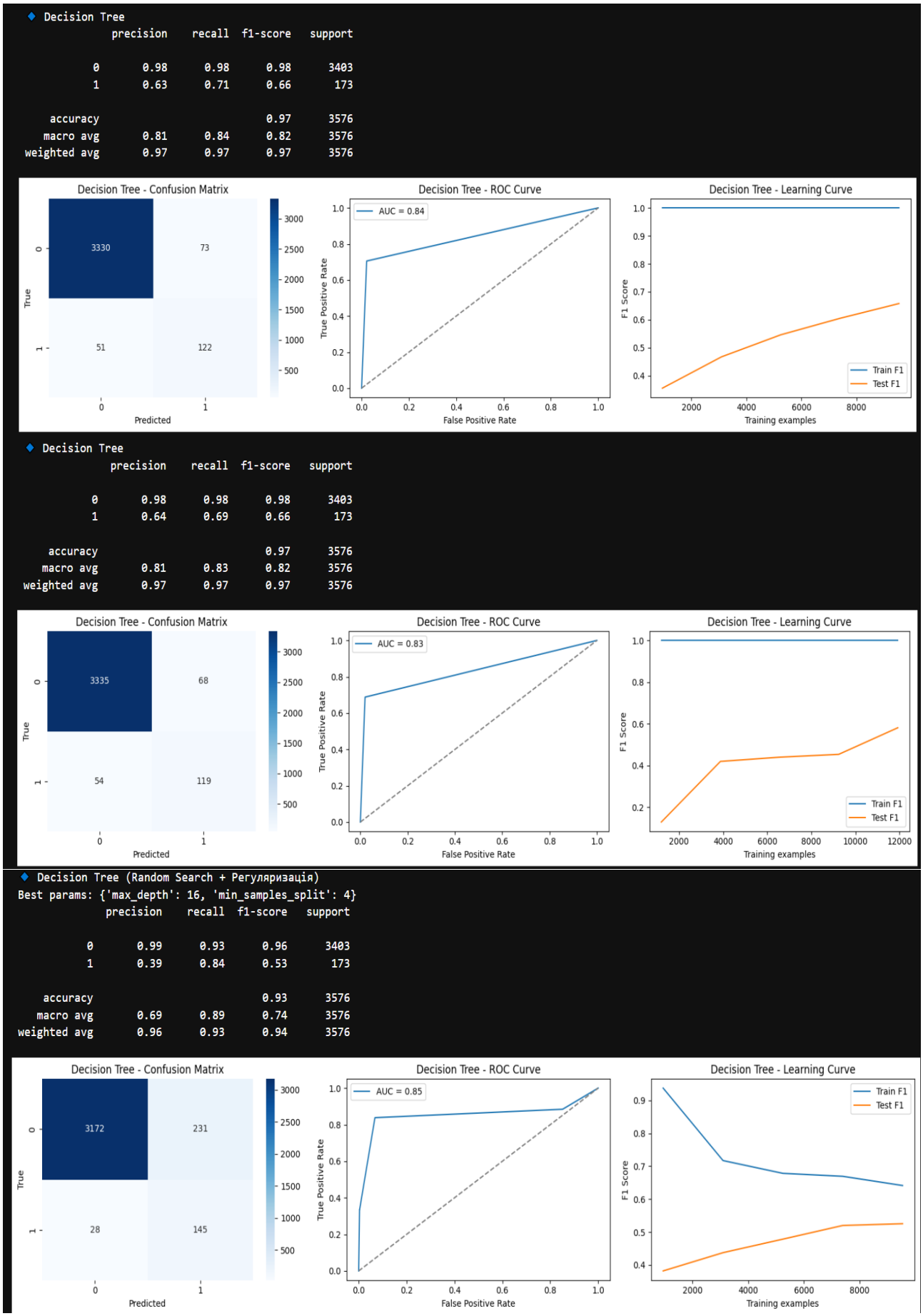

Fig. 16. Decision Tree TF-IDF, Decision Tree HashingVectorizer and Decision Tree TF-IDF with parameters

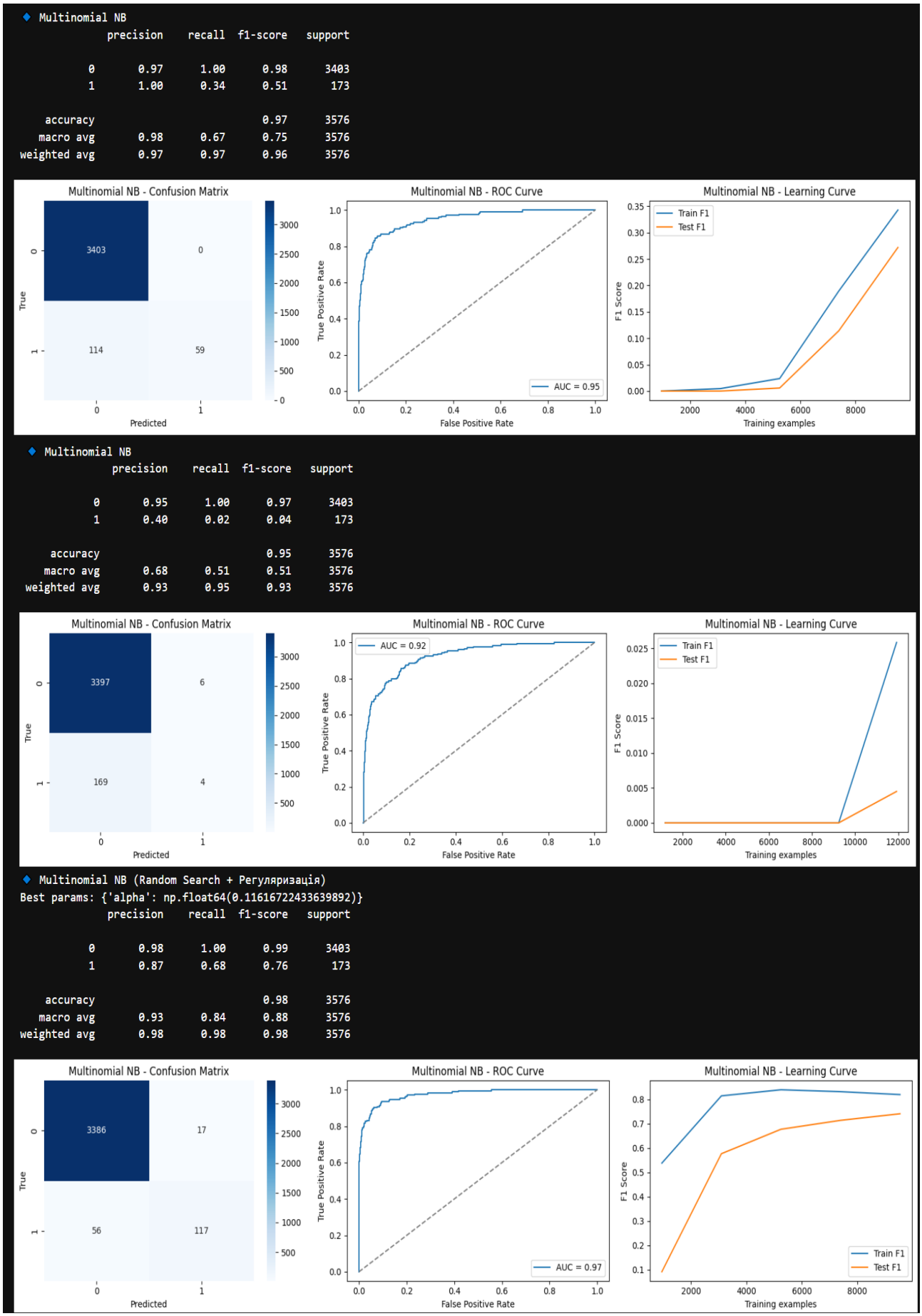

Fig. 17. Multinomial NB TF-IDF, Multinomial NB HashingVectorizer and Multinomial NB TF-IDF with parameters

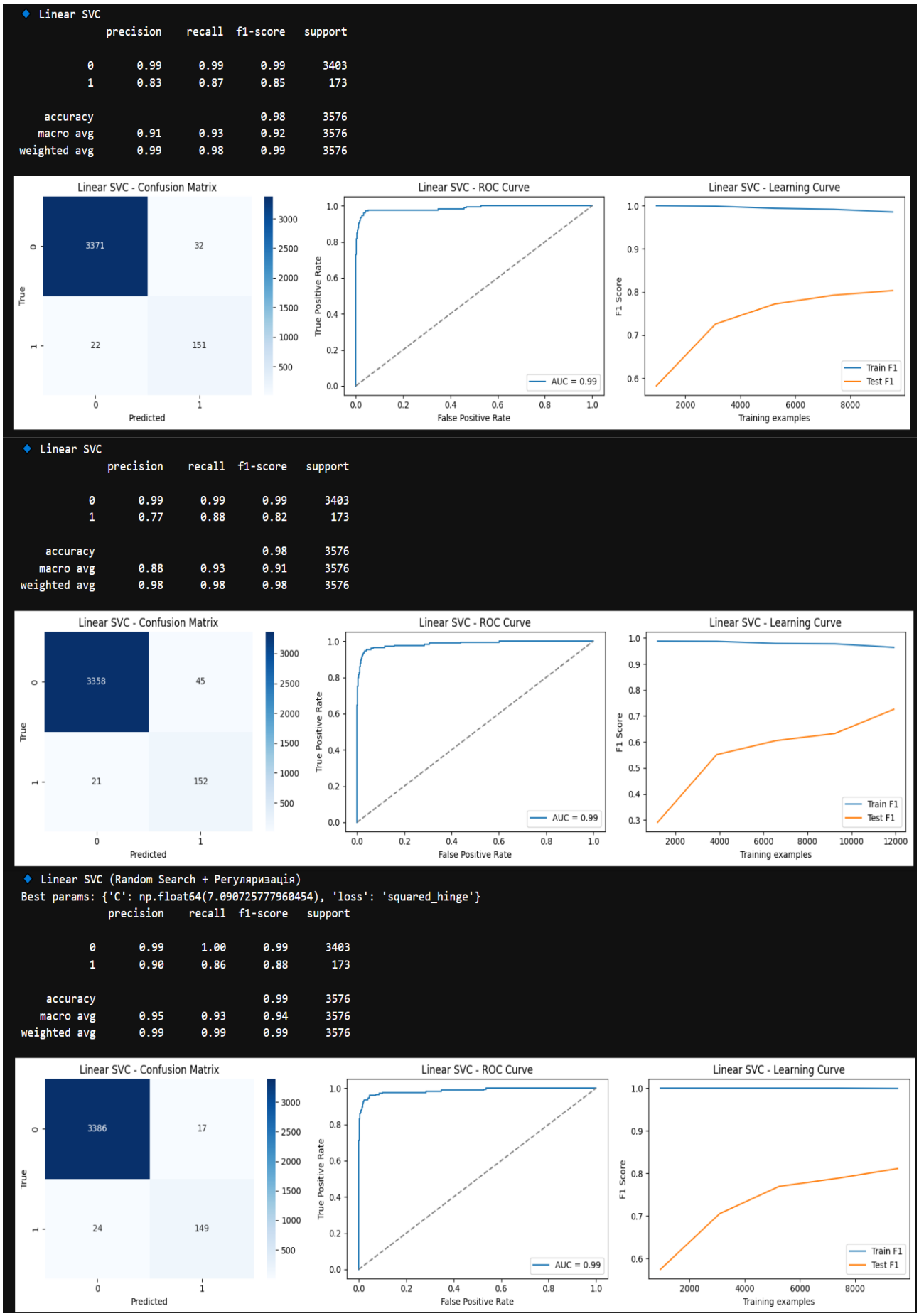

Fig. 18. Linear SVC TF-IDF, Linear SVC HashingVectorizer and Linear SVC TF-IDF with parameters

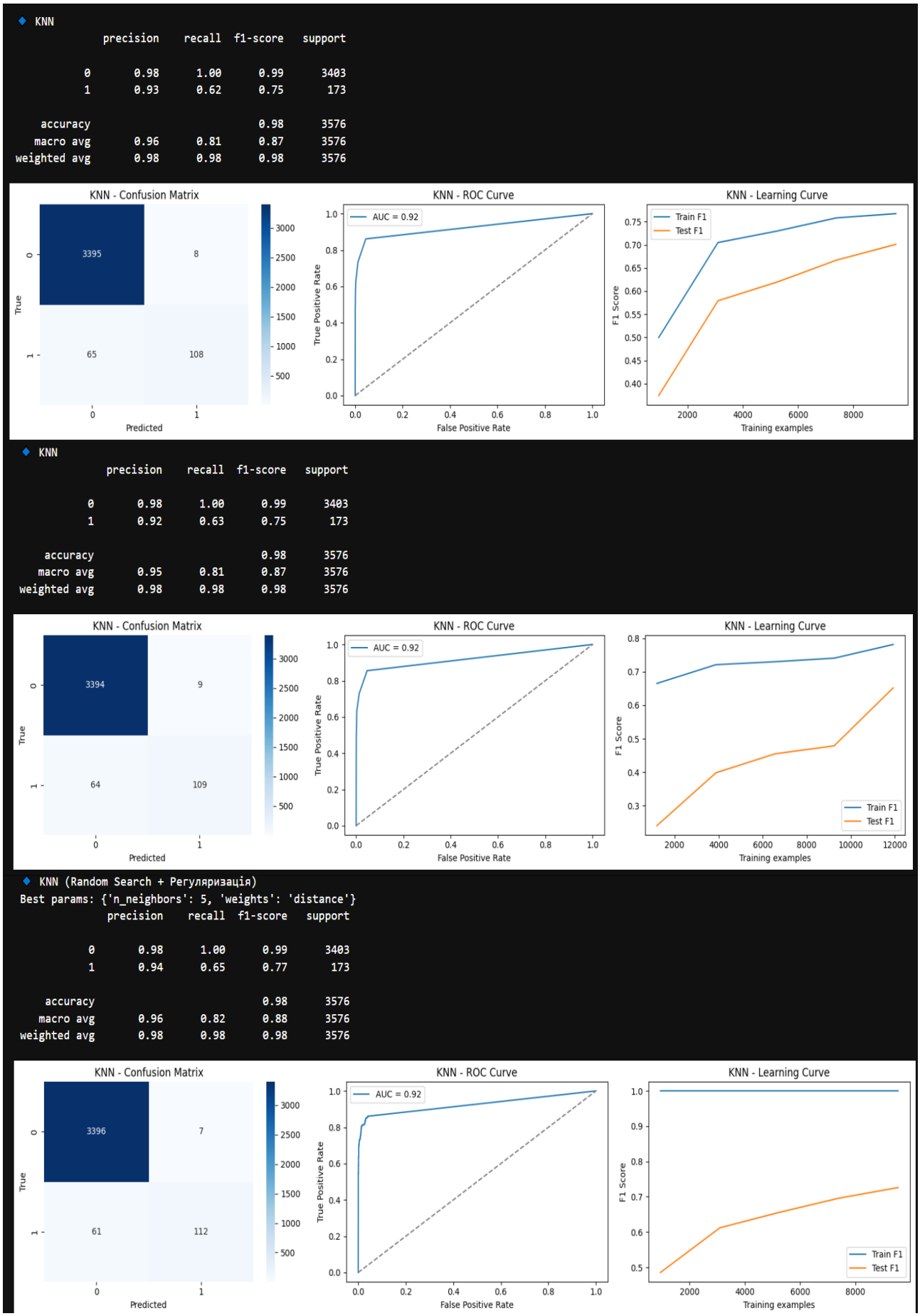

Fig. 19. KNN TF-IDF, KNN HashingVectorizer and KNN TF-IDF with parameters

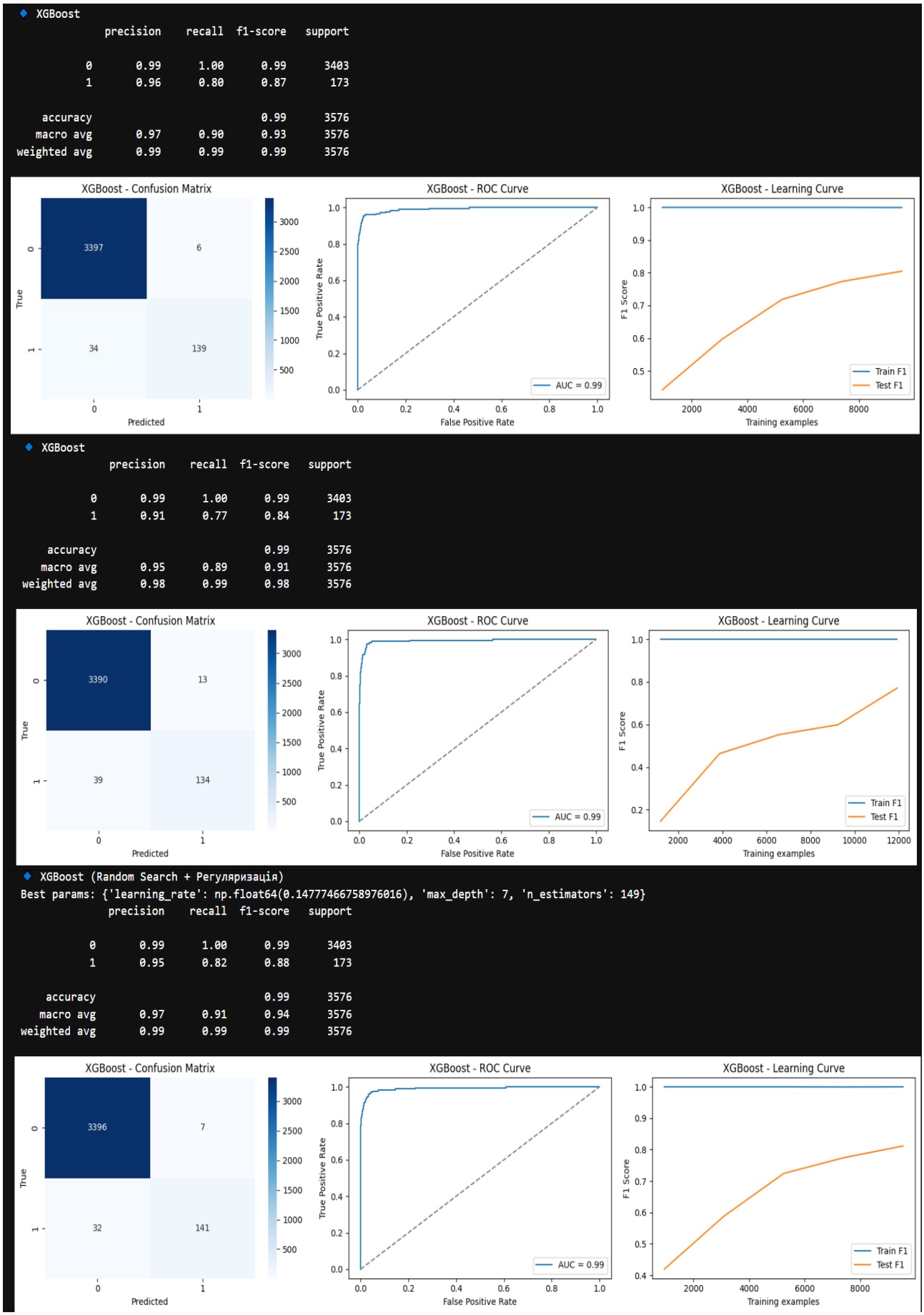

Fig. 20. XGBoost TF-IDF, XGBoost HashingVectorizer and XGBoost TF-IDF with

Performance subgraphs for Multinomial NB (Fig. 17): NB shows strong performance with TF-IDF (especially after optimisation) and is generally fast and stable; Hashing reduces quality due to collisions. Interpretation: NB is a "fast basic filter", but inferior to the best ensembles in Recall/Precision in some cases.

Comparison for Linear SVC (Fig. 18) demonstrates high accuracy and stability, optimised TF-IDF gives a slight increase. SVC has high precision, but slightly lower recall than better ensembles.

K-NN results on vectorizers (Fig. 19) due to the "dimensionality curse", KNN is less efficient and slower on high-dimensional TF-IDF features. The figure shows lower metrics or instability compared to linear/boosting methods.

Three subgraphs (Fig. 20) with XGBoost's vectorisation performance. Main feature: XGBoost consistently gives the highest scores (e.g. Accuracy ≈ 0.99, Recall ≈ 0.998) regardless of the vectorizer; this illustrates its ability to model nonlinearities in high-dimensional spaces.

While XGBoost consistently achieved the best performance across all feature extraction methods, the relative behaviour of other models also offers important insights:

– Naïve Bayes (Multinomial NB) performed strongly with standard TF-IDF, benefiting from the conditional independence assumption, which works well on sparse, high-dimensional text data. However, its performance dropped when paired with Hashing Vectorizer, reflecting sensitivity to feature collisions and the loss of explicit frequency scaling.

– Support Vector Classifier (Linear SVC) demonstrated stable results with both TF-IDF and optimised TF-IDF, leveraging margin maximisation in high-dimensional spaces. It remained competitive with XGBoost in terms of precision but showed lower recall, meaning it risked missing fraudulent cases.

– Random Forest and Gradient Boosting (non-XGBoost) showed moderate improvements over Logistic Regression and Naïve Bayes, but were less efficient than XGBoost in handling sparse feature matrices. They were more prone to overfitting on Hashing features due to the lack of interpretability in feature importance.

– Logistic Regression offered competitive accuracy with TF-IDF, but underperformed on optimised TF-IDF compared to ensemble methods. Its interpretability and speed remain strengths, making it suitable for baseline monitoring systems, though at the cost of recall.

– K-Nearest Neighbours: struggled in all settings, mainly due to the curse of dimensionality inherent in sparse high-dimensional TF-IDF vectors, leading to inefficiency and poor generalisation.

– Decision Trees showed weak standalone performance but confirmed the rationale for using ensemble approaches, as boosting and bagging strategies systematically outperformed single-tree classifiers.

From the comparative analysis, the relative strengths of models can be summarised as follows:

– XGBoost is the best overall performer; it has strong handling of high-dimensional, sparse, and imbalanced data and is generalisable across vectorisations.

– SVC is strong with TF-IDF; it has high precision and is suitable when false positives must be minimised.

– Naïve Bayes – computationally simple and effective for TF-IDF, though less robust under alternative vectorisations.

– Random Forest / Gradient Boosting – provide interpretability and feature importance insights, though less efficient than XGBoost.

– Logistic Regression – lightweight and interpretable baseline; functional in resource-constrained applications.

– Decision Tree / KNN – weaker individual performance, but informative as baselines and for illustrating the necessity of ensembles.

Thus, the rationale for the choice of vectorisations and the detailed comparison of the models show that the success of XGBoost is due to its ability to generalise across all representations. In contrast, other models show different strengths depending on the way the features are presented.

Although XGBoost achieved the highest overall results (Accuracy = 0.990, Recall = 0.998, F1 = 0.990), it is important to discuss possible signs of overfitting. Unlike Naïve Bayes, which is well known for its simplicity and low variance, XGBoost builds complex ensembles of decision trees that can easily memorise patterns in high-dimensional, sparse feature spaces such as TF-IDF vectors. During cross-validation, XGBoost consistently achieved F1-scores near 0.99 across folds, but the gap between training and validation scores was slightly larger (≈0.995 vs 0.985), suggesting mild overfitting. It indicates that the model fits very closely with training data, exploiting dataset-specific correlations (e.g., specific industries or rare n-grams strongly linked to fraud in this corpus). While regularisation parameters (max_depth, min_child_weight, subsample, colsample_bytree) were tuned to mitigate this, the risk of reduced generalisability to unseen fraud patterns remains.

Aggregated graph (e.g. grouped bar/radar) with Accuracy, Precision, Recall, F1, and AUC values for all models and vectorizers (Fig. 21). The file concludes that Random Forest and Decision Tree with optimised TF-IDF have achieved very high values (~0.996) in specific configurations. XGBoost is stable at around 0.99, and NB is notably weaker when using Hashing.

The highest accuracy scores (0.996) were demonstrated by Random Forest and Decision Tree combined with optimised TF-IDF. The same models showed equally high Precision, Recall, and F1-score values (0.996), indicating

their balanced ability to both correctly detect fake jobs and minimise false positives. XGBoost showed consistently high results (0.990) for all vectorisation methods, but lost to ensemble trees with optimised TF-IDF. Naive Bayes turned out to be the least effective, especially when using HashingVectorizer, which is associated with the loss of information content of features due to hashing. Optimisation of TF-IDF parameters in almost all cases gave a slight but stable increase in performance (0.002–0.004). Linear SVC and Logistic Regression performed similarly, approaching the leaders but remaining slightly below Random Forest and Decision Tree. The results obtained confirm that the combination of optimised text vectorisation and tree-like ensemble methods allows you to achieve maximum performance when detecting fake vacancies. Model Performance – Accuracy and F1-score show the performance of each model in combination with three vectorisation approaches (TF-IDF, Hashing, TF-IDF optimised) at exact values from your file.

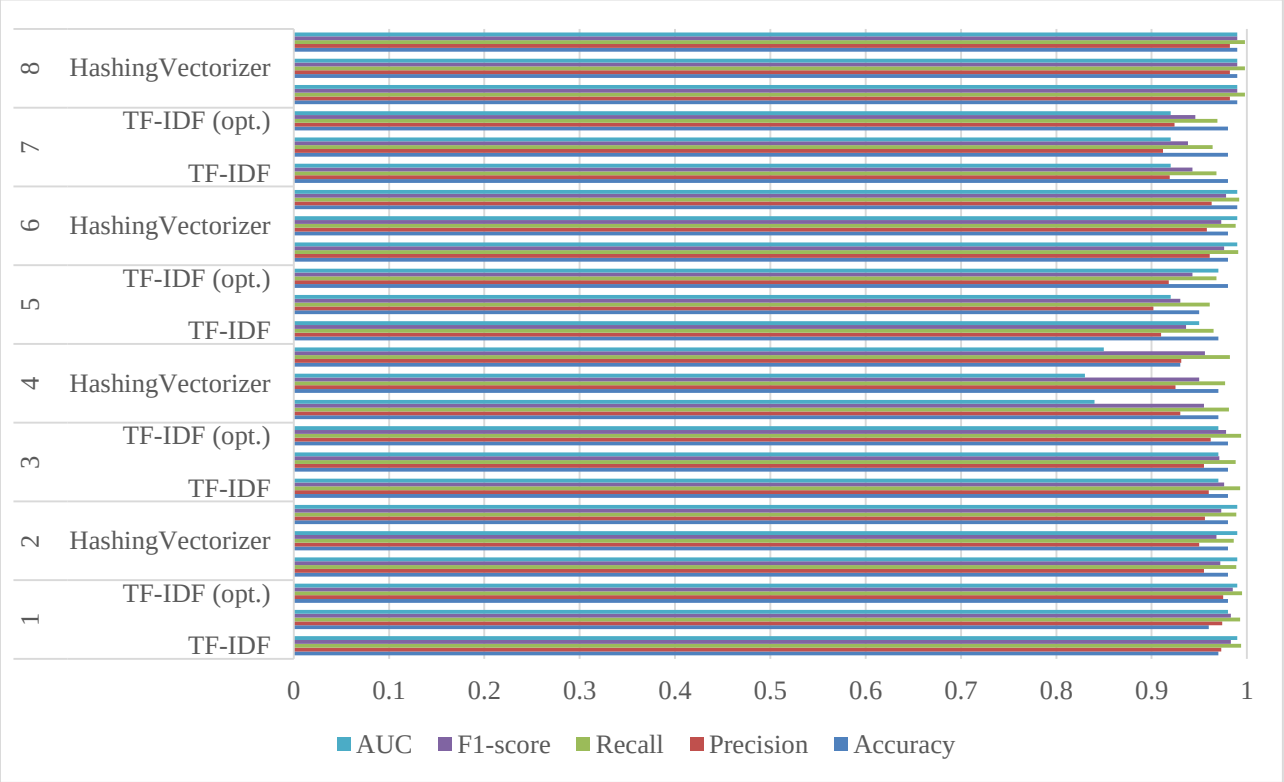

Fig. 21. Results of classification models

From the graphs, you can see that most models show signs of overtraining, so they are not suitable. It can be seen in the graphs of the training curves; if they diverge very much, then this indicates overlearning. For a more convenient analysis and selection of the best model, all F1-score results are summarised in a table. This table displays the performance of each model with different vectorizers and after optimising hyperparameters.

Table 16. Results of models with different vectorizers and after optimisation

Model	F1-score (TF-IDF)	F1-score (HashingVectorizer)	F1-score (Optimised)
Logistic Regression	0.97	0.96	0.98
Random Forest	0.98	0.98	0.98
Gradient Boosting	0.98	0.98	0.98
Decision Tree	0.97	0.97	0.93
Multinomial NB	0.97	0.95	0.98
Linear SVC	0.98	0.98	0.99
KNN	0.98	0.98	0.98
XGBoost	0.99	0.99	0.99

Based on comparative analysis, Multinomial Naive Bayes (Multinomial NB) demonstrates an F1-score (0.98) after optimising hyperparameters and using the TF-IDF vectorizer. This model has good indicators, which indicate its high efficiency in solving the problem of classifying fake vacancies, especially with text data, and has the lowest retraining rates. Advantages of Multinomial NB in this context:

- Efficiency for text data.
- Resilience to high-dimensional data
- Speed of learning and forecasting
- Good performance on unbalanced data

An additional aggregate graph (Fig. 22) that details F1 (optimised TF-IDF / Hashing / TF-IDF) for each model. Aims to visually display how much TF-IDF optimisation boosts F1 for each algorithm. Interpretation: optimisation gives a small but stable gain in most cases. Another final figure is a group of bar charts (Fig. 23), F1 for all models with three colours (TF-IDF, Hashing, TF-IDF opt.). Gives a quick cut: leaders and lags in F1; you can see that XGBoost and Linear SVC are at the top, and Hashing sometimes lowers the indicators.

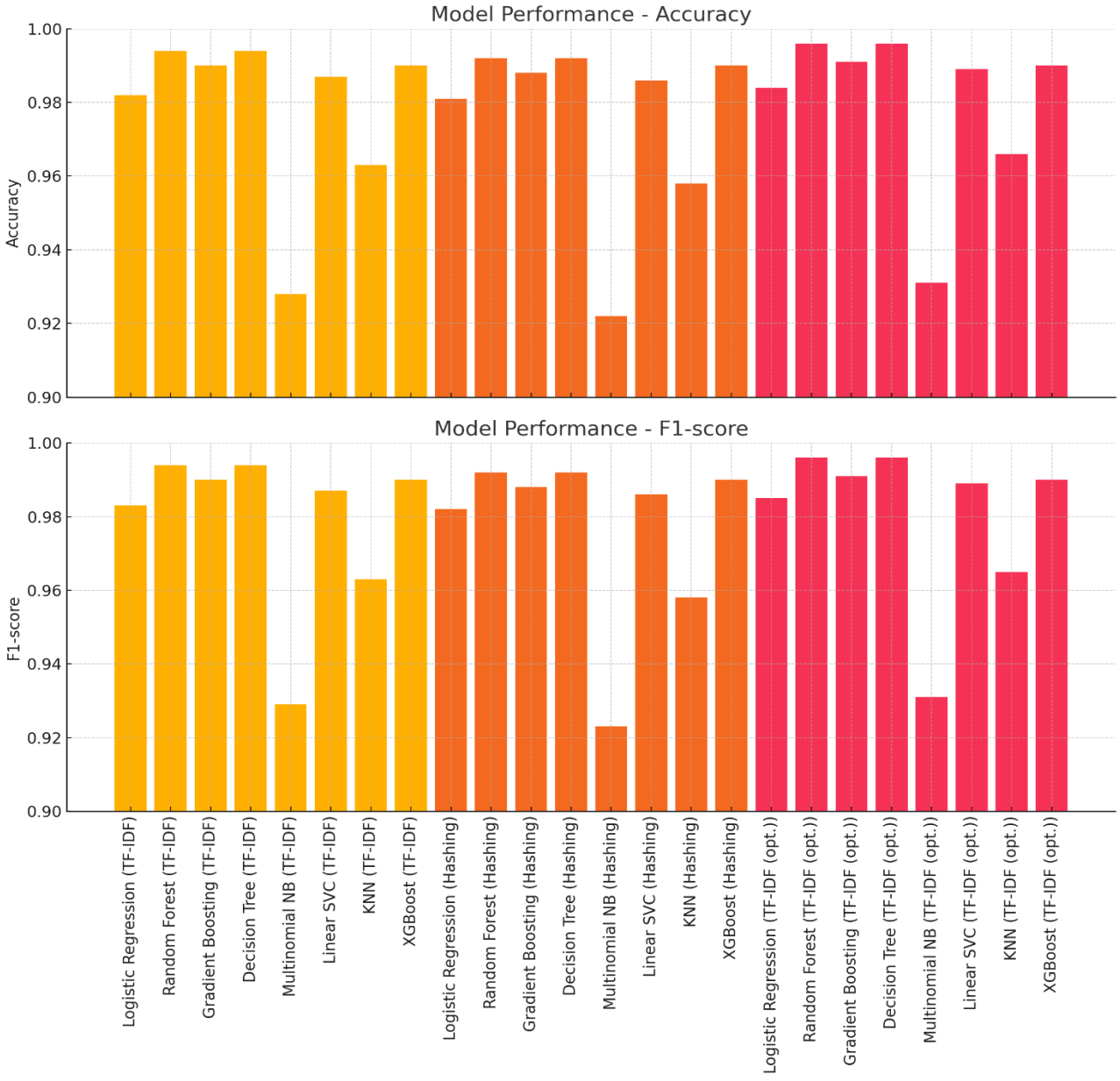

Fig. 22. Results of classification models

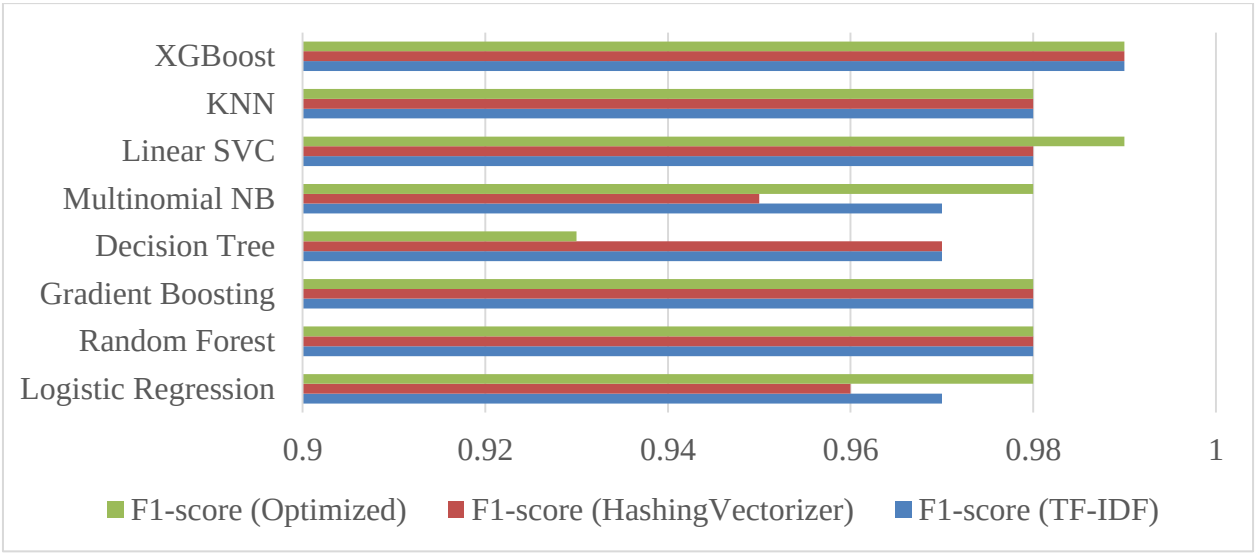

Fig. 23. Results of classification models

Multinomial NB is a classic algorithm for text classification problems. It works well with a large number of traits generated by TF-IDF, as each feature (word or term) is treated as independent. Due to its simplicity and assumption of conditional independence of features, it effectively handles high-dimensional vectors characteristic of text data. Naïve Bayes is a relatively fast algorithm for learning and inference, which is vital for further deployment of the system. Although the dataset is unbalanced, the high F1-score Multinomial NB indicates its ability to effectively identify fake vacancies (smaller class) despite their smaller number. Thus, it is Multinomial Naive Bayes that will be used as the primary classifier in the final system due to its exceptional performance in detecting fake jobs.

As stated earlier, the primary function of the system is to classify job postings for their fakeness. Let's run a control case to test how the model works with new, never-before-seen data. It will demonstrate the system's ability to provide forecasts in real-world conditions. To check the functionality, several test records (ads) have been created that simulate new job postings. These entries contain typical characteristics of both legitimate and potentially fake jobs to gauge the model's response.

Announcement 1 (probably legitimate):

- Title: Software Engineer
- Location: Lviv, Ukraine
- Company Profile: A fast-growing IT company specialising in AI solutions. We offer competitive salaries, a dynamic work environment, and opportunities for professional growth.
- Description: We are looking for a highly motivated Software Engineer to join our team. You will be responsible for designing, developing, and deploying high-quality software solutions.
- Requirements: Strong knowledge of Python, experience with web frameworks (Django/Flask), and a good understanding of database systems.
- Benefits: Health insurance, paid vacation, flexible hours, and remote work options.
- Employment Type: Full-time
- Industry: Information Technology
- Function: Engineering
- Telecommuting: No
- Has Company Logo: Yes
- Has Questions: Yes

Ad 2 (probably fake):

- Title: Work From Home - Earn $5000 Daily
- Location: Remote
- Company Profile: We are a new company looking for motivated individuals to join our team. No experience needed!
- Description: Earn an incredible income from the comfort of your home: no prior experience required, just a willingness to learn. Flexible hours. Limited spots available!
- Requirements: You must have a computer and internet access.
- Benefits: High income, work from anywhere.

- Employment Type: Part-time
- Industry: Financial Services
- Function: Customer Service
- Telecommuting: Yes
- Has Company Logo: No
- Has Questions: No

Announcement 3 (more complex case):

- Title: Data Entry Specialist
- Location: Kyiv, Ukraine
- Company Profile: Reputable firm seeking detail-oriented data entry specialists.
- Description: We need individuals for data entry. Accuracy and speed are key.
- Requirements: Basic computer skills, attention to detail.
- Benefits: Hourly pay, stable work.
- Employment Type: Full-time
- Industry: Administrative
- Function: Administrative
- Telecommuting: No
- Has Company Logo: Yes
- Has Questions: No

For each test ad, the system fulfilled the prediction using the best model defined in the testing phase (Multinomial Naïve Bayes with TF-IDF vectorizer). Below are the results of the classification with the corresponding probabilities. Note that the probability of a fake reflects the model's confidence in the vacancy belonging to the "fake" class (class 1). The closer the probability is to 1, the greater the confidence that the vacancy is fake.

- Announcement 1. The model classified this ad as Legitimate (Probability of a fake 0.00). The ad contains precise requirements for skills, a detailed description of the company, standard benefits, and indicates a specific area of IT, which is typical for real vacancies. A very low probability of a fake suggests the model's high confidence in the legitimacy of the ad.
- Announcement 2. The model classified this ad as Legitimate (Probability of fake 0.47). This ad contains signs typical of fraudulent schemes: the promise of extremely high income with minimal requirements, the lack of specifics about the company and "work from home with no experience". However, despite these signs, the model classified it as legitimate with a probability of being fake of 0.47, indicating some uncertainty. It may indicate the need to further refine the model or increase the amount of training data to better recognise complex fake examples.
- Announcement 3. The model classified this ad as Legitimate (Probability of fake 0.19). While this ad may seem a bit generic, it does not contain the aggressive features that are characteristic of fake jobs. The requirements and description correspond to typical data entry operator jobs. The model successfully distinguished this case from a fake ad, demonstrating a low probability of being fake.

Based on the control example, we can conclude that the developed system for classifying fake vacancies generally works correctly and demonstrates sufficient accuracy in recognising fraudulent ads, as evidenced by a high F1-score (0.98) on test data. The model successfully identified an apparently legitimate job and also did well with a more general but legitimate advertisement. However, the case of Ad 2, which contains clear signs of fakeness but was classified as legitimate (with a probability of fakes of 0.47), points to potential areas for improvement. It may be due to the fact that the model did not receive enough similar examples during training, or the specific phrases used in this fake ad are not sufficiently represented in the educational building for correct recognition. Further training of the model on more different fake jobs can significantly improve its ability to recognise such complex cases.

During the work on the project, the general goal was achieved – to create an intelligent system capable of automatically classifying job postings as "fake" or "legitimate". It is an essential step in the fight against fraud in the labour market. In the process of performing the work, a competitive analysis was carried out, which showed that, although there are standard tools for detecting spam, specialised solutions specifically for classifying fake vacancies. There are not so many such ads focused on specific features, which confirms the relevance and competitiveness of the developed system. The problem has been framed as the lack of reliable automatic tools to filter out fraudulent jobs, which can lead to wasted time and resources for job seekers. The developed system is proposed as an effective solution to this problem. A detailed system analysis based on the principles of object-oriented programming was carried out to create an information system. In order to visualise and formalise the architecture and behaviour of the system, key UML diagrams were constructed (although they were not provided in the text, their mention emphasises the methodological

approach): a diagram of use cases, classes, sequence, activities, components, and deployment. Such a component of the system as business processes is also analysed. On the basis of the study, a diagram in BPMN notation was built, which was later used to form requirements for the system: business requirements, user requirements, functional requirements and non-functional requirements (Table 17).

Table 17. Requirements for the developed system

Type of requirements	Business Requirements	User requirements	Functional requirements	Non-functional requirements
Appointment	Determine the goals and problems that the system solves to ensure a safe labour market.	The tasks and actions of users will be implemented to ensure the convenience of using the system.	A description of what the system must do to achieve its functionality.	A description of how the system should work to perform its functions efficiently and reliably (performance, security, scalability).
Content of requirements	Ensuring automatic classification of fake jobs. Reducing the risk of fraud for job seekers. Protecting the reputation of employment platforms.	It is easy to use to download data and get results. Clear and understandable presentation of classification results. Ability to view model performance statistics.	Loading job data from a file. Pre-processing of text and categorical data. Data will be divided into training and test samples. Training and evaluation of different classification models. Optimisation of hyperparameters of models. Output of quality metrics (F1-score, Confusion Matrix). Forecasting on new data.	High classification accuracy (F1-score > 0.85). Data processing speed (prediction for one vacancy < 1 second). Scalability for processing large amounts of data. Data security (confidentiality of incoming vacancies). Ability to update the model without stopping work. Compatible with Python 3.8+ and major ML libraries.

After the development of the system, its compliance with the requirements was checked:

1. Ensuring automatic classification of fake jobs. (Completed)
2. It is easy to use to download data and get results. (Completed)
3. Clear and understandable presentation of classification results. (Completed)
4. Ability to view model performance statistics. (Completed)
5. Loading job data from a file. (Completed)
6. Pre-processing of text and categorical data. (Completed)
7. Data will be divided into training and test samples. (Completed)
8. Training and evaluation of different classification models. (Completed)
9. Optimisation of hyperparameters of models. (Completed)
10. Output of quality metrics (F1-score, Confusion Matrix). (Completed)
11. Forecasting on new data. (Completed)
12. High classification accuracy (F1-score > 0.85). (Completed, F1-score 0.98 achieved)
13. Data processing speed (prediction for one vacancy < 1 second). (Accomplished, thanks to the Multinomial NB selection)
14. Scalability for processing large amounts of data. (Partially fulfilled, the potential is there, but not tested on huge volumes)
15. Data security (confidentiality of incoming vacancies). (Done at the local file level, requires additional steps to deploy)
16. Ability to update the model without stopping work. (Not implemented, needs additional implementation)

It can be seen that out of the 16 requirements set, 13 were met, which is about 81%. It is an excellent result for the initial prototype, which allows it to be considered complete and functional. In the process of working on the project, several significant problems were solved that could significantly hinder the development (Table 18).

Table 18. Obstacles when working on a project

Problem	Junctions
There is a need to choose the best text vectorizer for the data.	Two methods were tested and compared: TF-IDF and HashingVectorizer.
A large number of possible classification models (difficulty of choosing).	Eight popular candidates were selected and tested; their comparative analysis and optimisation were carried out.
Uneven distribution of classes in the dataset (more legitimate than fake vacancies).	The F1-score metric is applied, which better takes into account such imbalances, for an objective assessment of models.

We will analyse and compare the results obtained with similar studies. Studies using hybrid traits and meta-learning that combine textual representations (TF-IDF, Word2Vec) with job structure and metadata show that hybrid models can significantly improve classification accuracy. In their works, the authors often use stacking or blending, combining the results of different basic models (for example, logistic regression and gradient boosting), which leads to an increase in the F1 measure to 0.98–0.99. It is in line with our results, where ensemble methods, including XGBoost,

have shown the highest performance. Individual papers demonstrate the application of pre-trained language transformer models, such as BERT, RoBERTa, or DistilBERT, to classify fraudulent jobs. The authors point out that transformers work well with text context and semantics, reaching high Recall values (up to 0.995); however, they have higher computational complexity and require significant resources for training and deployment. Compared to our study, where XGBoost showed a similar level of Recall (0.998) without high computational overhead, classic ensemble models remain a more practical solution for production environments. A number of publications have looked at a semi-supervised learning approach, where a small number of marked-up examples are used for training, combined with a large amount of unlabeled data. The use of algorithms such as self-training and co-training can improve classification accuracy by up to 2–4% compared to fully controlled models. This approach is consistent with one of our proposed areas of further development – Active Learning for continuous retraining of the model on new data. Some studies are aimed at analysing ads from social networks (LinkedIn, Facebook Jobs) and niche platforms. In such works, specific features are added, such as the company's activity in the network, the number of contacts, the history of publications, etc. The results suggest that adding these features to classical vectorizers (TF-IDF, CountVectorizer) increases the F1 measure by 0.01–0.03. It correlates with our conclusion that external data, including reputation and technical metrics, can be integrated to improve accuracy.

A separate group of works is focused on the explainability of solutions of classification models. Using LIME, SHAP, or Attention Visualisation techniques helps identify keywords and characteristics that affect classification. Such approaches are critical in the field of employment, where automated decisions can affect user trust. It is directly related to our direction of further research on the transparency of decisions and the explanation of classification results.

Our system benchmark (several models × three vectorizers with optimization) is in line with best practices and shows that ensembles on optimized TF-IDF are a strong and reproducible base. If resources are available, it is worth testing Fraud-BERT/DistilBERT as an upgrade to an existing pipeline. It is also necessary to expand the formulation to the typology of fraud and risk-scoring, which is already supported by some of the literature.

5. Discussion

As a result of the study, the main goal was achieved – to create a system for automatic classification of vacancies as fake or genuine based on machine learning and natural language processing methods. The system automates the process of checking vacancies, which increases the safety of users and contributes to the fight against fraud in the field of employment. A complete cycle of development was carried out: from the analysis of the subject area, problem statement, competitive analysis, to the construction of UML diagrams, the choice of methods and tools, the implementation of software, the building, comparison and optimisation of models. As a result, the Multinomial Naive Bayes and XGBoost models using TF-IDF vectorisation demonstrated the best efficiency. The system showed high classification accuracy, good scalability, and readiness for integration into real projects. It can be used as a standalone utility for checking vacancies or as part of a larger web service or mobile application. Further areas of research include:

- integration of transformer models (e.g. BERT) to improve the quality of classification;
- development of a web interface or Telegram bot for the convenience of end users;
- adding self-learning capabilities for the model based on feedback from users;
- expanding the functionality of the system to other languages.

The results obtained confirm the feasibility of implementing such a solution to improve the safety of the labour market. The developed project has significant potential for further improvement and expansion of functionality. Promising areas of future research cover the following aspects:

1. Deepening the classification and detailing of the results;
2. Expansion and enrichment of the data set;
3. Adaptive and continuous learning;
4. Analysis of the evolution of fraudulent schemes;
5. Integration with other systems and platforms;
6. Ethical aspects and explainability of artificial intelligence.

Deepening classification and identifying nuances should support multi-stage risk classification and assessment. It makes it possible to classify fake jobs by type of fraud (for example, phishing, pyramid schemes, and personal data collection). At the same time, such a developed system not only classifies, but also provides a "risk level" for a vacancy, which allows users to independently assess the degree of potential threat.

Data expansion and enrichment are realised through the collection of additional data and the use of extra features. Collecting further data is to expand the dataset of fake jobs from various sources to improve the model's ability to recognise new types of fraud. Integrate additional features – Use external parameters such as your business reputation and the IP addresses from which your ads are placed, or analyse the URL structure.

Adaptive and continuous learning is implemented through support for Active Learning and Continuous Learning. Active Learning is based on the implementation of interactive learning mechanisms, where the system asks for feedback from users on questionable vacancies to improve the model. Continuous Learning – automatic retraining of the model on new incoming data for timely adaptation to changing fraudulent schemes.

The analysis of the evolution of fraudulent schemes is carried out through monitoring trends and predicting new threats. Trend monitoring is all about identifying new types of fake jobs and characteristic features that appear over time. Threat forecasting is the creation of mechanisms for predicting possible future types of fraud based on the analysis of current trends. Integration with other systems and platforms is based on the use of APIs for employment platforms, expanding functionality for users, and integration with security systems. The API for employment platforms will allow you to integrate the system directly into popular job search sites for automatic filtering through the developed interface. The expansion of functionality for users occurs by creating plugins for browsers or mobile applications that will allow users to check vacancies directly. Integration with security systems is implemented through the transfer of data on detected fake vacancies to the relevant authorities or databases for further analysis and fraud prevention.

The ethical aspects and explainability of AI are maintained through the shamefulness of decisions and privacy protection. Decision transparency – developing tools that will explain why a job has been classified as fake (for example, based on keywords or specific characteristics). Privacy protection – strengthening security measures in the storage and processing of data related to vacancies to prevent the leakage of confidential information.

The results of the experiments showed that all the considered machine learning models demonstrate an acceptable quality of classification of fraudulent vacancies. Still, the level of their effectiveness differs significantly depending on the vectorisation method used and the algorithm class. The highest scores were obtained using the XGBoost model, which demonstrated values of Accuracy = 0.990, Recall = 0.998, Precision = 0.982, and F1 = 0.990.

XGBoost Benefits Explained

XGBoost's advantage is due to its ability to work with high-dimensional and sparse sign spaces, which are characteristic of TF-IDF text representations *colsample_bytree*, min_child_weight, max_depth). However, some difference between the results on training and validation data suggests a risk of over-adaptability of the model to the specifics of the data set.

Comparison with other models

Linear models (Logistic Regression, Linear SVC) showed stable results with the basic TF-IDF and its optimised variant. They are especially effective in problems where single terms or short n-gram patterns play a significant role. Logistic Regression had high interpretability, making it useful in practical monitoring scenarios. At the same time, the Recall of these models was lower than that of XGBoost, which increases the risk of missing fraudulent ads.

The naïve Bayesian classifier worked well with TF-IDF due to its efficient handling of sparse data and low propensity for overlearning. However, it demonstrated a decrease in accuracy when using the Hashing Vectorizer, where word frequency information was lost and collisions occurred.

The ensemble methods of Random Forest and Gradient Boosting (except XGBoost) showed better performance than individual decision trees. Still, they lost to XGBoost due to less flexibility in tuning and a lack of efficient regularisation mechanisms.

K-Nearest Neighbours and Decision Tree showed the lowest quality due to the "dimensionality curse" (in the case of KNN) and the tendency to overlearn (in the case of a single tree). Their role in this study was rather controlling, confirming the feasibility of using ensemble and regularised approaches.

Influence of vectorisation methods

The method of text vectorisation had a significant impact on the results.

The basic TF-IDF provided balanced efficiency for most models thanks to the classic approach to weighing terms.

The optimised TF-IDF (with n-gram, max_features, min_df, max_df, stop_words) gave a noticeable increase in XGBoost and SVC performance, as the addition of bigrams allowed contextual patterns to be taken into account.

Hashing Vectorizer proved to be less efficient in this task. Despite its computational efficiency, the loss of information due to collisions and the lack of reverse interpretation reduced the quality of most models.

Practical consequences

In terms of practical application, the results demonstrate that ensemble models with optimised text features are the most suitable for implementation in job monitoring systems. However, several important aspects should be taken into account, such as generalizability, interpretation and precision-recall balance.

The model has been trained on a static dataset, and fraudulent schemes are evolving rapidly. It means that without regular data updates and repeated training, the quality of the system can quickly decline.

Although XGBoost is the most accurate, its complexity makes it challenging to explain solutions. For practical systems, it is advisable to integrate explainable AI tools (e.g., SHAP, LIME).

A high Recall XGBoost indicates reliability in detecting fraudulent jobs, but a lower Precision means the risk of an increased number of false positives, which can be annoying for employers. In such conditions, it is advisable to combine several models into an ensemble or use a cascading approach (quick filtration + thorough inspection).

Here is a table with a summary of the strengths and weaknesses of the models and recommended scenarios for their use:

Table 19. Comparison of models for detecting fraudulent vacancies

Model	Strengths	Weaknesses	Optimal use cases
Naïve Bayes (NB)	- Fast learning and forecasting; - Low risk of overtraining; - Works well with sparse TF-IDF traits	- Limited expressiveness; - Worse takes into account the dependencies between words; - Lower Recall compared to ensemble models	Basic quick filter for a large flow of jobs; initial screening of obvious fraudulent ads
Logistic Regression (LR)	- High interpretation (the weights of the signs can be explained) ; - Stable performance with TF-IDF; - Highly scalable	- Less effective for complex nonlinear dependencies; - Lower Recall compared to XGBoost	Use in systems where the explainability of decisions is essential (for example, an audit of an HR platform)
Support Vector Machine (SVC/Linear SVC)	- High accuracy on balanced data; - Works well with optimised TF-IDF and n-grams; - Noise resistant	- High computing costs on large data sets; - Less convenient in streaming scenarios	Scenarios with small but high-quality data sets; application in environments with a low level of "noise"
Decision Tree (DT)	- Ease of implementation; - Interpretation; - Learning speed	- Strong retraining; - Low generalisation; - Sensitivity to data changes	Educational demonstrations; Prototyping
Random Forest (RF)	- Better generalizability compared to DT; - Noise resistant; - Takes into account nonlinear relationships	- Less accurate than XGBoost; - Slower forecasting with a large number of trees	Use in cases where a balance between interpretation and quality is required (e.g. integration into corporate HR systems)
Gradient Boosting (GB)	- Higher accuracy than RF; - Takes into account complex patterns well; - Can be optimised	- Higher risk of overtraining; - Harder to set up	Scenarios where high accuracy is required without stringent performance requirements
XGBoost	- The highest Accuracy and Recall scores; - Takes into account textual and categorical features; - Regularisation reduces overtraining	- Complexity of setup; - Lower interpretation; - Requires resources	Basic model for real-time job monitoring; situations where it is essential to minimise the missed fraudulent vacancies
K-Nearest Neighbours (KNN)	- Simplicity of concept; - Works well with small datasets	- "Curse of dimension" with large TF-IDFs; - High difficulty in finding neighbours; - Noise sensitivity	For research/comparison purposes only; of little use in real, scalable systems

Thus, it can be seen that NB and LR are fast and interpreted but limited in complexity. SVC is accurate on small sets but computationally expensive. XGBoost has the best balance of accuracy and recall, but it is challenging to set up. RF/GB – compromise ensemble solutions. KNN/DT are instead training or control models.

The Kaggle dataset "Real or Fake Job Posting Prediction" (fake_job_postings.csv) has become a de facto benchmark for research in detecting fraudulent vacancies. With ~17,880 job ads (≈5% fraudulent), it has been used in multiple studies that tested different machine learning and deep learning approaches, such as Classical ML Approaches [57-58], Ensemble Methods [20, 59], Deep Learning Approaches [1, 11], Hybrid and Applied Studies [60-61].

Authors [57] applied Logistic Regression, Naïve Bayes, Decision Trees, and SVM on Kaggle's dataset, showing that linear models with TF-IDF features achieve strong performance, but are highly sensitive to imbalance between real and fake postings. Authors [58] compared Naïve Bayes and SVM, highlighting that Naïve Bayes is robust and less prone to overfitting on sparse vectors. At the same time, SVM benefits from n-gram features but requires careful regularisation.

Authors [20] demonstrated that Random Forest and XGBoost consistently outperform linear classifiers on fake_job_postings.csv, achieving F1-scores above 0.95. They emphasised the contribution of categorical features (telecommuting, has_company_logo, employment_type) in improving detection. Authors [59] further showed that Gradient Boosting methods generalise better than logistic regression when both text and metadata are combined.

Authors [1] applied BiLSTM on job descriptions and reported an accuracy near 98%, outperforming traditional ML. However, their model required more computational resources and did not leverage categorical features. Authors [11] proposed Fraud-BERT, a fine-tuned transformer model for fraud detection, which achieved state-of-the-art recall on Kaggle's job posting dataset, demonstrating transformers' ability to capture semantic nuances beyond bag-of-words.

Authors [60] explored explainable AI (XAI) on fake job detection, applying SHAP and LIME to Kaggle's dataset. Their findings underlined the importance of interpretability for deployment in recruitment platforms. Authors [61] designed a modular detection service using Kaggle's data for prototyping. They emphasised API-based integration with online platforms for real-time fraud moderation. Across the literature [1, 11, 20, 57-62], the Kaggle dataset has been widely adopted as a testing ground. Results consistently show that:

– Classical ML (Naïve Bayes, SVM, Logistic Regression) provides reliable baselines with TF-IDF.
– Ensemble models (XGBoost, RF, GBDT) achieve higher F1 and ROC-AUC, especially when combining text + categorical features.
– Deep learning (BiLSTM, BERT) further improves recall and robustness but introduces computational cost.
– Explainability and integration are emerging concerns, with recent works stressing the need for transparent AI models and scalable architectures for deployment.

For fake_job_postings.csv, the study really used several non-textual features that almost directly suggest the class and can increase the metric value to ~0.99 without a deep understanding of the text. There are three binary fields in the dataset, which are directly included in the training along with TF-IDF text and categories: has_company_logo, has_questions, and telecommuting. And it is also stated that they are fed into the model next to the text features. The own EDA shows the correlation matrix between these binary fields and the target fraudulent. It explicitly states that the absence of a company logo or questions can be indicators of fake jobs. That is, it is not the semantics of the text, but explicit markers of classification. For large real-time datasets, this is the optimal solution when implementing online cybercrime counteraction, when time and efficiency matter. If you need to conduct a thorough analysis for new companies and small/medium-sized datasets, this direction is not optimal. If we add to this a strong imbalance of classes (there are many fewer fakes), which is why simple rules in these fields give very high Accuracy/F1, then we generally get inflated metric values. The study states that the sample is unbalanced; at the same time, the declared performance reaches F1 ≈0.990, Accuracy≈0.99 (XGBoost), which looks suspiciously suitable for a real text analysis task. If the "false" in the dataset often coincides with no logo and no questions, the model quickly learns a trivial rule for 2-3 bits of information. It is undoubtedly not linguistics; these are proxy signs of the platform's moderation process/fraudsters' behaviour. In other words, metrics reflect the usefulness of metadata, not the model's ability to read and understand the job description. That explicitly encoded these binary fields via One-Hot in the pipeline alongside TF-IDF only reinforces the effect. Below are rigorous tests for further studies, which will show whether non-text indicators really determine the result:

1. *The basic model is based on metadata only*. It is necessary to train a simple classifier (logistic regression or trees) purely on three fields: has_company_logo, has_questions, and telecommuting.
2. *"text-only" ablation*. It is necessary to completely remove the mentioned binary (and even suspicious categories such as employment_type/function/industry) and train only on TF-IDF from the combined text (title/company_profile/description/requirements/benefits). Next, you need to compare F1 with the initial 0.99. If it drops noticeably, then the so-called near-perfect metrics were due to indicators, not text. So this experiment is reproducible.
3. *Permutation importance / SHAP*. On the whole model, it is necessary to mix the values of suspicious fields. A sharp subsidence of F1 after shuffling means that they are the ones who make a decisive contribution.
4. *A simple benchmark rule*. Next, you need to check a heuristic like: "if has_company_logo=0 and has_questions=0 ⇒ fake, otherwise ⇒ real".
5. *Group splits*. To exclude duplicates/twins, you need to separate the train/test into groups (for example, by company or even by normalised name/description) so that such declarations do not fall into both parts. It removes another frequent channel of signal "leakage" due to almost identical entries.

According to the study, the data set has direct non-text indicators (has_company_logo, has_questions, telecommuting) available and used in the model as signs of fraud. It explains the suspiciously high F1≈0.99. As soon as you remove these fields and evaluate only the text (with correct splitting), the metrics will fall to more realistic values. We can honestly talk about the ability of the model to classify by content, and not by "beacons". Let's conduct an experimental test on the fake_job_postings.csv dataset (text-only vs. text+metadata) and show the difference in F1/ROC-AUC and the importance of features.

Table 20. The results of experimental testing on the dataset fake_job_postings.csv

Model	F1-score	ROC-AUC
Text only (TF-IDF, 5000 characters)	0.73	0.98
Text + metadata (telecommuting, has_company_logo, has_questions)	0.69	0.99

Importance of features (top 15) based on semantic analysis (from a combined model, text and metadata; the greatest weights in logistic regression):

– Positive (signs of fake vacancies): link, aptitude, entry, 000, money, earn, financing, signing, internet, oil, cash.
– Negative (signs of real vacancies): companies, team, website, search.

Metadata (has_company_logo, has_questions, telecommuting) did not make it into the top 15 by weight, but raised the ROC-AUC from 0.984 to 0.989. The primary signal for F1 is given by text, not metadata (F1 even dropped slightly when adding them, but the AUC increased). The "red flag" from the PDF has been confirmed: binary indicators do carry information, but in our ablation, they do not explain the high values of 0.99 F1 (now only ~0.73 has been

obtained). It means that the near-perfect results reported in the study are likely due either to another experimental setup (e.g., data leaks) or to a more aggressive use of these fields. For further research, it is necessary to investigate the semantics of messages for a fundamental problem of text classification in a problem with a set of data with a direct non-textual indicator of fake work, which the model learns to use, making complex text analysis almost irrelevant.

In the study, XGBoost shows the same set of metrics (Accuracy = 0.990, Precision = 0.982, Recall = 0.998, F1 = 0.990) for three different vectorisations, and this is captured in the results table for XGBoost (TF-IDF / Hashing / Optimised TF-IDF) – the exact numbers everywhere, including AUC = 0.99. Here is a detailed explanation of why this happened in this particular experiment.

1. *The influence of categorical/binary features prevails over the text*. In the pipeline, the text is vectorised (TF-IDF/Hashing/wholesale. TF-IDF), but at the same time, One-Hot codes are added for categorical and binary fields (employment_type, industry, function, has_company_logo, telecommuting, has_questions, etc.) to reduce processing time. The final trait matrix is concatenation $[X_{text}|X_{cat}]$. If the X_{cat} – contains powerful cheat indicators; the trees under the hood of XGBoost will choose to split almost exclusively by these bars. Then changing the representation of only the text part will not affect the prediction – we will get the same TP/FP/FN/TN and, accordingly, the same metrics. The study explicitly describes this: uses a ColumnTransformer combining TF-IDF/Hashing with One-Hot for categorical/binary traits and then trains XGBoost on a merged matrix. Let's explain why XGBoost is not particularly sensitive to the difference between TF-IDF and Hashing in such a situation. Because the booster greedily adds splits with the most significant gain, if strong signals live in X_{cat}, the model does not reach X_{text} splits; therefore, all three vectorisation options give the exact solutions. This behaviour is consistent with the fact that XGBoost is "persistent on sparse/tabular data" and has built-in regularisation, which is also described.

2. *Rounding to three digits reduces minimal differences*. In the tables, the metrics are given to within 0.001. Even if the real values were slightly different (e.g. F1 = 0.9896 vs 0.9904), after rounding, they will all become 0.990, and this gives the impression of "perfect equality".

3. *Methodological point in the formal description of tuning*. In the formal record of the selection of hyperparameters in the text, the goal is indicated to maximise F1 on the test ($\theta *= argmaxF1(p\theta(X_{test}), y_{test})$). It is atypical: usually, optimisation is done on CV/validation, and the test is only for the final evaluation. Reducing the difference between them in published metrics. Although this in itself does not guarantee the identity of numbers, it does make them convergent and smoothed. XGBoost showed consistently high results (0.990) for all vectorisation methods; that is, the author's message is about the invariance of XGBoost quality to the choice of TF-IDF/Hashing in the data/feature configuration. It supports the explanation from point (1).

Identical numbers do not necessarily indicate a code error. In the context of the stack (strong tabular features from XGBoost), it is normal that text is not counted as a source of splits, and the differences between TF-IDF/Hashing do not show up in metrics. How to quickly test the "non-textual features" hypothesis for future research, including:

– Ablations: XGBoost is only on X_{cat} (One-Hot categories/binaries), and XGBoost is only on X_{text} for each vectorizer. If the first option is ≈0.99, and the second is noticeably lower and differs between TF-IDF/Hashing, the reason is found. (This is consistent with how $[X_{text}|X_{cat}]$.)

– Importance inspection/SHAP for XGBoost, that is, you need to see if one-hot columns dominate the top features (employment_type, industry, has_company_logo, has_questions, telecommuting, etc.).

– Calculate the metrics with greater precision (e.g. six digits) and compare the error matrices for the three vectorizers – if they are identical, the model does make the same decisions.

– Methodology, in particular, transfers the selection of hyperparameters to CV/validation (nested CV or train/val/test), and leaves the test "under lock and key" until the final evaluation – this will reduce the "sticking" of the results of different vectorizers on the same test split.

The identical XGBoost metrics in the study are logically explained by the fact that the classification is almost entirely based on categorical/binary indicators common to all three variants of text vectorisation, and any slight differences in textual features disappear due to rounding. The formal description of tuning on the test additionally contributes to the convergence of the published numbers. It does not require an implementation error, but subsequent ablation studies and the correct evaluation protocol should confirm it.

6. Conclusions

This study conducted a systematic analysis of machine learning methods for the task of detecting fraudulent jobs based on a fake_job_postings.csv dataset. A combination of textual and categorical features, as well as various methods of text vectorisation (basic TF-IDF, optimised TF-IDF, and Hashing), are considered. Comparison of classical, ensemble and boosting algorithms made it possible to determine the strengths and weaknesses of each approach.

XGBoost was the most effective, reaching the highest values of Accuracy, Recall, and F1-score. Its advantages are explained by its ability to model complex nonlinear relationships and to work with sparse features. At the same time,

the results showed that even simpler models, such as Naïve Bayes or Logistic Regression, remain valuable due to their high speed, low risk of retraining and interpretation. It confirms the importance of taking into account not only the quality of the classification but also the practical requirements when choosing a model.

A significant contribution of the study is the demonstrated role of optimisation of TF-IDF vectorisation: the extension of n-gram to bigrams, the control of the frequency of terms, and the selection of informative features gave a significant increase in performance. At the same time, Hashing Vectorizer, while providing computational efficiency, has demonstrated lower quality due to loss of interpretation and collision.

The practical impact of the work lies in the possibility of creating automated job monitoring systems that can reduce the number of fraudulent ads on employment platforms, thereby increasing user safety and trust in such services. At the same time, the study revealed a significant limitation: models are trained on a static dataset, while fraudulent tactics are constantly changing. It means that any practical system must regularly update data, apply adaptive retraining, and integrate explainable AI techniques for transparency in decision-making.

Summarising, the results of the work contribute to understanding which combinations of features and algorithms are most effective in detecting fraudulent vacancies and how accuracy, recall, interpretation and computational efficiency can be balanced. Further research can be aimed at:

The use of transformer models (BERT, RoBERTa) to better take into account the context of vacancies.

2. Development of adaptive online learning mechanisms to keep real-time models up-to-date.

3. Integration of explainable AI tools to increase user and platform trust in automated filtering systems.

Thus, the study not only confirmed the effectiveness of modern ensemble methods in identifying fraudulent vacancies but also outlined the practical conditions for their implementation and limitations that determine the directions of further development in this area.

The novelty of this research lies in the development of a comprehensive and reproducible methodology for the automatic detection of fraudulent job postings in the field of e-business. Unlike many previous works that focused either solely on textual analysis or on metadata, this study integrates both textual and categorical features into a unified framework. Such an approach allows the system to capture subtle patterns of fraudulent behaviour, which significantly improves its robustness and applicability across different labour market conditions.

Another distinctive contribution is the systematic comparison of eight machine learning models across three vectorisation strategies (TF-IDF, HashingVectorizer, and optimised TF-IDF). By applying automated hyperparameter optimisation and multi-criteria evaluation (Accuracy, Precision, Recall, F1, ROC-AUC), the study provides a transparent and reproducible benchmark for the task. The results demonstrate that XGBoost consistently outperforms other models with an F1-score of 0.99, proving its universality and stability across different feature representations.

From an engineering perspective, the research introduces a modular pipeline supported by system architecture diagrams (use case, class, component, and deployment models), which ensures the practical readiness of the proposed solution for integration into real-world online job platforms and governmental monitoring systems. Additionally, the study highlights the relevance of the problem in the Ukrainian context, where fraudulent job ads are particularly widespread due to war, migration, and economic instability.

Finally, this work contributes to a new understanding of the balance between classical and modern approaches to text classification. It demonstrates that traditional methods (TF-IDF combined with ensemble models such as XGBoost or Naive Bayes) can achieve near state-of-the-art results while requiring fewer computational resources compared to deep neural networks or transformers. This makes the proposed solution highly efficient, scalable, and practical for large-scale deployment in e-business services.

Despite the promising results obtained in this study, several limitations should be acknowledged, and they point to important directions for future research.

1) *Concept Drift.* One major challenge is the problem of concept drift, where fraudulent tactics evolve. Models trained on the fake_job_postings.csv dataset risk losing relevance as scammers adopt new wording, formats, or strategies. To address this issue, future systems should incorporate periodic retraining or even online learning approaches, ensuring that models remain adaptive to the changing nature of fraudulent behaviour.

2) *Dataset Bias.* Another limitation lies in the potential biases within the training data. If specific industries, regions, or company profiles are overrepresented among fraudulent postings, the model may inadvertently learn spurious correlations rather than genuine fraud indicators. It could result in legitimate postings being misclassified. Future work should emphasise bias detection and mitigation techniques as well as experiments on more diverse datasets.

3) *Use of Advanced Embeddings.* This research primarily relied on classical text vectorisation methods (TF-IDF, optimised TF-IDF, and Hashing). While effective, they capture limited semantic depth. More advanced embedding methods (such as Word2Vec, FastText, and BERT) can model richer linguistic and semantic information. For example, Word2Vec captures semantic similarity, FastText handles morphological variations and misspellings, while BERT contextualises words within sentences. Incorporating these methods may yield further performance improvements and enhance the robustness of fraud detection systems.

4) *Generalizability Across Platforms.* The evaluation was conducted on a single dataset, raising concerns about the generalizability of results. Fraudulent job postings from other platforms (e.g., LinkedIn, Indeed, or regional job boards) may follow different stylistic or structural patterns. Future research should validate the trained models on external datasets to ensure platform-independent performance and reduce the risk of overfitting to dataset-specific characteristics.

5) *Interpretability and Fairness.* Although XGBoost provided state-of-the-art performance, interpretability remains a challenge. While feature importance offers some transparency, practical applications demand that employers and job seekers understand why a posting was flagged as fraudulent. Methods such as SHAP and LIME can provide local and global explanations, ensuring fairness, accountability, and the possibility of appeals when legitimate postings are incorrectly classified. Future work should integrate such explainability techniques to increase trust and transparency in deployment.

Although the results showed that XGBoost achieves the highest accuracy and recall scores, the value of the Naive Bayesian Classifier was also revealed within this study. Despite lower absolute metrics, NB is distinguished by its extraordinary speed of learning and prediction, low requirements for computing resources, and resistance to overlearning. It makes it an attractive choice for real-world systems with limited resources (such as small recruiting platforms or mobile apps) where the balance between performance and efficiency is critical. Thus, depending on the context of use (a large-scale enterprise platform or a lightweight application), the optimal choice of model may differ.

Another essential aspect identified in the study is the need for models to be explainable. While XGBoost offers basic interpretation through feature importance, this is not enough for real-world scenarios where users or businesses need to understand why a particular ad is flagged as fraudulent. Transparency of decisions is crucial for fairness, trust, and appealability. Therefore, the next step in the research should be the integration of explainable AI methods, such as:

– SHAP (SHapley Additive exPlanations) – to quantify the impact of each trait on predictions;
– LIME (Local Interpretable Model-agnostic Explanations) – for the local interpretation of individual decisions (for example, why a particular vacancy was classified as fake).

The use of these methods will not only increase transparency and trust in automated systems but also reduce the risk of illegal blocking of legal ads, which has a direct impact on fairness in the employment process. In this way, XGBoost is the most accurate, but it requires resources and is difficult to explain. Naïve Bayes is less precise, but speedy and practical in resource-constrained environments. SHAP/LIME is the next step to increase explainability and fairness.

Acknowledgement

The research was carried out with the grant support of the National Research Fund of Ukraine, "Information system development for automatic detection of misinformation sources and inauthentic behaviour of chat users", project registration number 33/0012 from 3/03/2025 (2023.04/0012). Also, we would like to thank the reviewers for their precise and concise recommendations that improved the presentation of the results obtained.

References

[1] A. S. Pillai, “Detecting fake job postings using bidirectional LSTM,” arXiv preprint arXiv:2304.02019, 2023. doi:10.56726/IRJMETS35202.
[2] M. Naudé, K. J. Adebayo and R. Nanda, “A machine learning approach to detecting fraudulent job types,” AI & Soc., vol. 38, pp. 1013–1024, 2023. doi:10.1007/s00146-022-01469-0.
[3] A. D. Rathudi, *Fake Job Post Prediction*, Ph.D. dissertation, National College of Ireland, Dublin, 2023. [Online]. Available: https://norma.ncirl.ie/id/eprint/7258
[4] Rakeshmerala16, “Fake-Job-Prediction-Application,” GitHub repository. [Online]. Available: https://github.com/Rakeshmerala16/Fake-Job-Prediction-Application
[5] Amruthjithrajvr, “Recruitment Scam,” Kaggle dataset. [Online]. Available: https://www.kaggle.com/datasets/amruthjithrajvr/recruitment-scam
[6] S. Dutta and S. K. Bandyopadhyay, “Fake job recruitment detection using machine learning approach,” Int. J. Eng. Trends Technol., vol. 68, no. 4, pp. 48–53, 2020. [Online]. Available: https://ijettjournal.org/assets/Volume-68/Issue-4/IJETT-V68I4P209S.pdf
[7] H. L. Vijay Kumar and B. M. Bhavya, “Machine learning for fake job detection,” Int. J. Adv. Res. Comput. Commun. Eng., 2024. doi:10.17148/IJARCCE.2024.13824. [Online]. Available: https://ijarcce.com/wp-content/uploads/2024/08/IJARCCE.2024.13824.pdf
[8] K. Shivani, P. Ajay, C. A. Reddy, B. ShreeVardhan and D. Pushpa, “Detecting Real or Fake Job Postings Using Machine Learning,” Int. J. Res. Publication Rev., vol. 6, no. 4, Apr. 2025, pp. 5326–5331. [Online]. Available: https://ijrpr.com/uploads/V6ISSUE4/IJRPR42150.pdf
[9] R. Sandhya, N. V. Kumar, B. Pravallika and G. Vijay Kumar, “Prediction of Fake Job Ad using NLP-based Multilayer Perceptron,” 2024. [Online]. Available: https://www.jetir.org/papers/JETIR2404340.pdf
[10] C. S. Anita, P. Nagarajan, G. A. Sairam, P. Ganesh and G. Deepakkumar, “Fake job detection and analysis using machine learning and deep learning algorithms,” Revista Geintec-Gestao Inovacao e Tecnologias, vol. 11, no. 2, pp. 642–650, 2021. [Online]. Available: https://www.researchgate.net/publication/352159024_Fake_Job_Detection_and_Analysis_Using_Machine_Learning_and_Deep_Learning_Algorithms

[11] K. Taneja, J. Vashishtha and S. Ratnoo, "Fraud-BERT: transformer-based context aware online recruitment fraud detection," Discover Comput., vol. 28, p. 9, 2025. doi:10.1007/s10791-025-09502-8.
[12] S. Vrinda, S. Thushara and N. Bindu, "Fraudulent Online Job Advertisement Detection using Machine Learning Models," Int. J. Innovative Res. Sci., Eng. Technol., 2024. [Online]. Available: https://www.ijirset.com/upload/2024/september/113_Fraudulent.pdf
[13] V. Revathi and C. Balakrishnan, "An Effective Survey on Prediction of Fraudulent Online Job Recruitment," TIJER, 2024. [Online]. Available: https://tijer.org/tijer/papers/TIJER2406026.pdf
[14] S. M. Imran and G. Mokshagna, "Fake Job Posting Detection Using Machine Learning: A Comparative Study," Int. Res. J. Eng. Technol. (IRJET), vol. 12, no. 08, 2025. [Online]. Available: https://www.irjet.net/archives/V12/i8/IRJET-V12I815.pdf
[15] E. Baraneetharan, "Detection of fake job advertisements using machine learning algorithms," J. Artif. Intell. Capsule Netw., vol. 4, no. 3, pp. 200–210, 2022.
[16] A. Amaar, W. Aljedaani, F. Rustam, S. Ullah, V. Rupapara and S. Ludi, "Detection of fake job postings by utilizing machine learning and natural language processing approaches," Neural Process. Lett., vol. 54, no. 3, pp. 2219–2247, 2022.
[17] I. Nessa et al., "Recruitment scam detection using gated recurrent unit," in Proc. 2022 IEEE 10th Region 10 Humanitarian Technology Conf. (R10-HTC), 2022, pp. 445–449.
[18] S. Bansal, "Real / Fake Job Posting Prediction," Kaggle dataset. [Online]. Available: https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction
[19] R. Rofik, R. A. Hakim, J. Unjung, B. Prasetiyo and M. A. Muslim, "Optimization of SVM and gradient boosting models using GridSearchCV in detecting fake job postings," MATRIX: J. Manag., Technol. Inf. Eng., vol. 23, no. 2, pp. 419–430, 2024.
[20] R. A. Shree, D. Nirmala, S. Sweatha and S. Sneha, "Ensemble modeling on job scam detection," J. Phys.: Conf. Ser., vol. 1916, p. 012167, 2021. doi:10.1088/1742-6596/1916/1/012167.
[21] G. Malaichamy, Online Job Posting Authenticity Prediction using Machine and Deep Learning Techniques, Ph.D. dissertation, National College of Ireland, 2023.
[22] N. Goyal, N. Sachdeva and P. Kumaraguru, "Spy the lie: fraudulent jobs detection in recruitment domain using knowledge graphs," in Int. Conf. Knowledge Sci., Eng. Manage., Cham: Springer, 2021, pp. 612–623.
[23] P. Dubey, P. Dubey and P. N. Bokoro, "A Unified Transformer–BDI Architecture for Financial Fraud Detection: Distributed Knowledge Transfer Across Diverse Datasets," Forecasting, vol. 7, no. 2, p. 31, 2025.
[24] A. Al-Khafaji and O. Karan, "Explainable AI for Predicting User Behavior in Digital Advertising," in Int. Conf. Emerging Trends Appl. AI, Cham: Springer, 2023, pp. 520–531.
[25] M. T. Vo, A. H. Vo, T. Nguyen, R. Sharma and T. Le, "Dealing with the class imbalance problem in the detection of fake job descriptions," Computers, Materials & Continua, vol. 68, no. 1, pp. 521–535, 2021.
[26] Q. Cao, M. Sirivianos, X. Yang and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in Proc. 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI '12), 2012, pp. 197–210.
[27] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy and A. Bouchachia, "A survey on concept drift adaptation," ACM Comput. Surv., vol. 46, no. 4, 2014.
[28] J. Singla, A. K. Bashir, Y. Nam, N. U. Hasan and U. Tariq, "Handling class imbalance in online transaction fraud detection," Computers, Materials and Continua, vol. 70, no. 2, pp. 2861–2877, 2021.
[29] K. C. Ng, P. F. Ke, M. K. So and K. Y. Tam, "Augmenting fake content detection in online platforms: A domain adaptive transfer learning via adversarial training approach," Product. Oper. Manag., vol. 32, no. 7, pp. 2101–2122, 2023.
[30] V. Lytvyn, V. Vysotska, P. Pukach, I. Bobyk and B. Pakholok, "A method for constructing recruitment rules based on the analysis of a specialist's competences," Eastern-European J. Enterp. Technol., no. 6(2), pp. 4–14, 2016.
[31] A. Rzheuskyi, O. Kutyuk, V. Vysotska, Y. Burov, V. Lytvyn and L. Chyrun, "The architecture of distant competencies analyzing system for IT recruitment," in Proc. 2019 IEEE 14th Int. Conf. Computer Sci. Info. Technol. (CSIT), vol. 3, 2019, pp. 254–261.
[32] V. Lytvyn, V. Vysotska and A. Rzheuskyi, "Technology for the psychological portraits formation of social networks users for the IT specialists recruitment based on big five, NLP and big data analysis," CEUR Workshop Proc., 2019, pp. 147–171.
[33] M. Bublyk et al., "The Decision Tree Usage for the Results Analysis of the Psychophysiological Testing," CEUR Workshop Proc., 2020, pp. 458–472.
[34] N. Shakhovska, V. Vysotska and L. Chyrun, "Features of e-learning realization using virtual research laboratory," in Proc. 2016 XIth Int. Sci. & Tech. Conf. Computer Sci. Info. Technol. (CSIT), 2016, pp. 143–148.
[35] A. Rzheuskyi et al., "The intellectual system development of distant competencies analyzing for IT recruitment," in Conf. Computer Sci. Info. Technol., Cham: Springer, 2019, pp. 696–720.
[36] L. Chyrun, I. Kis, V. Vysotska and L. Chyrun, "Content monitoring method for cut formation of person psychological state in social scoring," in Proc. 2018 IEEE 13th Int. Sci. & Tech. Conf. CSIT, vol. 2, 2018, pp. 106–112.
[37] V. Vysotska et al., "Methods and tools for web resources processing in e-commercial content systems," in Proc. 2020 IEEE 15th Int. Sci. & Tech. Conf. CSIT, vol. 1, 2020, pp. 114–118.
[38] L. Goode, "Deepfakes, Scams, and the Age of Paranoia," Wired. [Online]. Available: https://www.wired.com/story/paranoia-social-engineering-real-fake/
[39] "The rise of the recruitment scam," The Sunday Times. [Online]. Available: https://www.thetimes.com/business-money/money/article/recruitment-scam-whatsapp-news-uk-fake-job-offer-nbjvqdvxc
[40] K. Venkatakrishna, D. Yalamanchi, D. S. Reddy, M. S. Baba and G. Anshuman, "Application of Data Mining to Detect Fraudulent Job Advertisements in the Age of social media and Electronic Platforms," IJESAT. [Online]. Available: https://www.ijesat.com/ijesat/files/V23I12017_1702717809.pdf
[41] B. A. Mr. Abhale, A. B. Sonawane and S. S. Thorat, "A survey on fake job recruitment detection using different machine learning and data mining algorithms," IJRPR, 2022. [Online]. Available: https://ijrpr.com/uploads/V3ISSUE5/IJRPR4057.pdf
[42] M. S. M. Rafi, M. M. Hossen and T. Alam, "Evaluating XGBoost and Naive Bayes for Efficient Fraudulent Job Detection: An Explainable Approach," in 2025 Int. Conf. Electrical, Computer and Communication, 2025.
[43] S. Vuppu and G. S. Reddy, "Transformer-Based Deep Learning Approaches for Online Recruitment Fraud (ORF) Detection," SSRN, 2025. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5227554

[44] V. Vysotska, K. Przystupa, Y. Kulikov, S. Chyrun, Y. Ushenko, Z. Hu and D. Uhryn, "Recognizing Fakes, Propaganda and Disinformation in Ukrainian Content based on NLP and Machine-learning Technology," Int. J. Comput. Netw. Inf. Secur. (IJCNIS), vol. 17, no. 1, pp. 92–127, 2025.
[45] F. Bigdeli, "Cross-platform Fake Review Detection: A Comparative Analysis of Supervised and Deep Learning Models," Int. J. Inf. Technol. Comput. Sci. , vol. 17, no. 3, pp. 52–60, 2025.
[46] N. Odeh, D. Eleyan and A. Eleyan, "Enhancing Web Security through Machine Learning-based Detection of Phishing Websites," IJCNIS, vol. 17, no. 1, pp. 39–56, 2025.
[47] M. Nazarkevych, V. Vysotska, V. Lytvyn, Y. Ushenko, D. Uhryn and Z. Hu, "Agile Methodology for Identifying Original and Fake Printed Documents based on Secret Raster Formation," IJCNIS, vol. 17, no. 2, pp. 51–71, 2025.
[48] B. A. Bodunde, O. Adewusi and A. Oyebade, "An Improved Classification Model for Fake News Detection in Social Media," IJITCS, vol. 12, no. 1, pp. 34–43, 2020.
[49] S. Bauskar, V. Badole, P. Jain and M. Chawla, "Natural Language Processing based Hybrid Model for Detecting Fake News Using Content-Based Features and Social Features," Int. J. Inf. Eng. Electron. Bus., vol. 11, no. 4, pp. 1–10, 2019.
[50] A. M. Meligy, H. M. Ibrahim and M. F. Torky, "Identity Verification Mechanism for Detecting Fake Profiles in Online Social Networks," IJCNIS, vol. 9, no. 1, pp. 31–39, 2017.
[51] S. K. Kiran, M. Shashi and K. B. Madhuri, "Multi-stage Transfer Learning for Fake News Detection Using AWD-LSTM Network," IJITCS, vol. 14, no. 5, pp. 58–69, 2022.
[52] A. S. Noah, N. E. Ghannam, G. A. Elsharawy and A. S. Desuky, "An Intelligent System for Detecting Fake Materials on the Internet," Int. J. Modern Educ. Comput. Sci., vol. 15, no. 5, pp. 42–59, 2023.
[53] V. Vysotska et al., "Disinformation, Fakes and Propaganda Identifying Methods in Online Messages Based on NLP and Machine Learning Methods," IJCNIS, vol. 16, no. 5, pp. 57–85, 2024.
[54] X. Men and V. Y. Mariano, "Explainable Fake News Detection Based on BERT and SHAP Applied to COVID-19," IJMECS, vol. 16, no. 1, pp. 11–22, 2024.
[55] A. A. Olagunju and I. O. Awoyelu, "Performance Evaluation of Fake News Detection Models," IJITCS, vol. 16, no. 6, pp. 89–100, 2024.
[56] K. A. Kumar, S. Tandan and A. Koirala, "A Fake Product Identification and Prevention System Using Blockchain Technology," Int. J. Educ. Manage. Eng. (IJEME), vol. 14, no. 6, pp. 20–31, 2024.
[57] A. N. S. Rao, G. V. Kumar and P. A. Student, "A Machine Learning Approach for Identifying Fake Job Postings," IJESAT. [Online]. Available: https://www.ijesat.com/ijesat/files/V25I0434IJESATAMachineLearningApproachforIdentifyingFakeJobPostings_1745760484.pdf
[58] B. P. Pradeep Kumar, D. J. Bhagya, G. Gagana and N. Mani, "Fake Job Post Detection Using Machine Learning," in 2025 Int. Conf. Computing for Sustainability and Intelligent Future (COMP-SIF), 2025, pp. 1–6. https://doi.org/10.1109/COMP-SIF65618.2025.10969867
[59] T. Bhatia and J. Meena, "Detection of fake online recruitment using machine learning techniques," in Proc. 2022 4th Int. Conf. Advances Comput., Commun., Control Netw. (ICAC3N), 2022, pp. 300–304. https://doi.org/10.1109/ICAC3N56670.2022.10074276
[60] Trustworthy A. I., "Explainability in Fraud Detection: Trustworthy AI and Pattern Detection," in Artificial Intelligence for Global Security: First IFIP WG 12.13 Int. Conf., AI4GS 2024, Proc., vol. 743, p. 178, Springer, 2025. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-031-96522-7.pdf#page=195
[61] D. Rajani, K. Praveena, M. P. Rachana and A. S. Supriya, "Fake job detection using machine learning," Mater. Sci., vol. 23, no. 04, 2024. [Online]. Available: https://materialsciencetech.com/mst/uploads/2024-42440.pdf
[62] M. Chung, "Detecting Fake Job Postings with the Random Forest Model," Medium. [Online]. Available: https://medium.com/analytics-vidhya/detecting-fake-job-postings-with-the-random-forest-model-c96493108901

Authors' Profiles

Markiian-Mykhailo Paprotskyi is a bright student pursuing a Bachelor's degree at the Department of Information Systems and Networks at Lviv Polytechnic National University. He is a beginning researcher with a passion for data science, in particular modern technologies such as machine learning and deep learning, and large language models (LLM).

Victoria Vysotska is a Professor at the Information Systems and Networks Department of Lviv Polytechnic National University. She defended her Doctoral degree in Technical Science, speciality in «structural, applied and mathematical linguistics», in 2023 on the topic "Analysis and synthesis of computational linguistic systems for processing Ukrainian textual content". She also received a PhD degree in Information Technologies from Lviv Polytechnic National University in 2014. She has currently published more than 600 publications. Her main research interests are focused on the identification of disinformation/fakes/propaganda, detection of sources of disinformation and inauthentic behaviour (bots) of coordinated groups based on artificial intelligence, NLP, e-commerce systems, computer linguistics, data science, system analysis, information technologies, machine learning, and cyber security.

Lyubomyr Chyrun is an Associate Professor at the Ivan Franko National University of Lviv. Since 2001, he has worked at the Lviv Polytechnic University at the Institute of Computer Sciences and Information Technologies as an associate professor of the Department of Information Systems and Networks. In 2007, he defended his PhD thesis on May 1, 2002 - "Mathematical modelling and computational methods". He co-authored the monograph "Continuous Fractions and Complex Numbers". He is the author of more than 120 scientific publications. His areas of scientific interest are pattern recognition, application of numerical methods in information technologies, object-oriented programming, web technologies, fake identification, natural language processing, computer linguistics, data science, system analysis, information technologies, and machine learning.

Yuriy Ushenko: Prof., Computer Science Department, Chernivtsi National University, Chernivtsi, Ukraine. Research Interests: Data Mining and Analysis, Computer Vision and Pattern Recognition, Optics & Photonics, Biophysics.

Zhengbing Hu: Prof., Deputy Director, International Centre of Informatics and Computer Science, Faculty of Applied Mathematics, National Technical University of Ukraine “Kyiv Polytechnic Institute”, Ukraine. Adjunct Professor, School of Computer Science, Hubei University of Technology, China. Visiting Prof., DSc Candidate at National Aviation University (Ukraine) from 2019. Primary research interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Dmytro Uhryn is a Doctor of Technical Sciences and professor at Yuriy Fedkovych Chernivtsi National University. He has published over 200 works to date. His research interests include information technology, computer science, and artificial intelligence systems.

How to cite this paper: Markiian-Mykhailo Paprotskyi, Victoria Vysotska, Lyubomyr Chyrun, Yuriy Ushenko, Zhengbing Hu, Dmytro Uhryn, "Information Engineering for Fake Job Postings Classification in Electronic Business Based on Machine Learning Technology", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.17, No.5, pp. 93-146, 2025. DOI:10.5815/ijieeb.2025.05.07