Combating the Surge in Fraudulent Job Advertisements: An Evaluation of Machine Learning Strategies on Data Balancing Techniques

DOBALLAH Mehdi (mehdo971)
Linköpings Universitet
TDDE16
Mehdo971@student.liu.se
Link to the code

Abstract

The number of fraudulent job ads postings has significantly increased in recent years. This activity can be motivated by various reasons such as collecting data to sell it, infiltrating companies for phishing, ransomware, or personal benefits. Several machine learning approaches have been attempted to classify ads for fake jobs. In our project, we are facing a problem: all the public datasets for this kind of research are imbalanced. Therefore, it will be imperative for us to use balancing systems (undersampling/oversampling) to see if they have a positive or negative effect on the model's performance. The performance of the models will be judged based on the AUC score, known for its precision in classifying binary problems.

Introduction

Data privacy today is more crucial than ever, especially when it comes to the security of personal information. Countless malicious individuals try their best to make money by selling this information to the highest bidder. One way to collect such sensitive information is by posting fake job ads. In this way, with the increase in the number of job seekers, they massively gather information to later sell. By creating fake job ads, it becomes clear which audience can be targeted, thus potentially causing even more damage with phishing for example

With the goal of combating the rising problem of fake job ads, researchers worldwide have attempted to develop machine learning models to distinguish between real and fraudulent job advertisements. The main issue encountered in developing such models is that the available datasets are extremely imbalanced, with real job postings far outnumbering the fraudulent ones. This imbalance leads to models being more biased towards the majority class, in this case, genuine job opportunities. In this paper, we aim to explore whether different oversampling/undersampling methods can improve the performance of a fraudulent job ad classifier.

1 Theory

1.1 Count Vectorizer

Machine learning algorithms inherently lack the capability to directly interpret text in its raw form. This necessitates the transformation of text into a format that these algorithms can process and analyze. The process involves vectorizing the text, where each word, or "token", is converted into a numerical representation. This conversion is typically carried out through two main steps: the fitting phase and the transform phase.

During the fitting phase, the algorithm scans a collection of documents to identify and assign a unique numerical index to every distinct token it encounters. This step essentially builds a vocabulary of all unique words found across the documents.

Subsequently, in the transform phase, each document is converted into a sparse matrix based on this vocabulary. In this matrix, rows represent individual documents, and columns correspond to the unique tokens identified during the fitting phase. The entries in this matrix indicate the frequency of each token within each document.

This method of text vectorization enables machine learning algorithms to effectively work with textual data by quantifying the presence and prevalence of words in a manner that's analyzable, allowing for further processing and insight extraction from text-based datasets.

1.2.1 Bernoulli Naïve Bayes

Naïve Bayes classifiers operate under the assumption that all features are conditionally independent, given the target class, which can be either win or loss (here 0 or 1/ fraudulent or non-fraudulent). The Bernoulli Naive Bayes is a part of this type of classifiers.

If we denote by x_i the i-th binary variable out of n binary variables, the likelihood of a document given a class Ck can be expressed as $\boldsymbol{p(X|C_k) = \prod_{i=1}^{n} p_{ki}^{x_i}(1-p_{ki})^{(1-x_i)}}$, where p_{ki} represents the probability of Class Ck generating the term x_i.

1.2.2 Logistic regression

The logistic model, a cornerstone in statistical analysis, offers a way to predict the likelihood of a particular event by relating the log odds of the event to independent variables through a linear relationship. Essentially, it employs a logistic function, encapsulated by the equation $\boldsymbol{p(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}}$, where x_i denotes the i-th independent variable among n variables, and p(x) symbolize the probability of the event's occurrence, influenced by these variables.

Central to this model is the sigmoid function, $\sigma(t) = \frac{1}{1+e^{-t}}$, which ensures output values are constrained between 0 and 1 (fraudulent and non-fraudulent for us). The logistic function essentially extends the sigmoid function to incorporate multiple independent variables, maintaining the output's range between 0 and 1, making it suitable for estimating probabilities.

Contrary to common perception, logistic regression itself does not categorize outcomes into distinct classes; instead, it calculates the probability of an event. Classification is achieved by setting a threshold: probabilities above this cutoff are assigned to one class, and those below to another. This binary classification mechanism enables its application in various fields, from medicine to ML, where it's crucial to predict categorical outcomes based on a set of predictors.

1.2.3 Decision Trees

Decision trees has a structure composed of numerous internal decision-making nodes. These nodes evaluates specific attributes to guide decisions (for instance, assessing cloudiness to predict rain). Each pathway emanating from a node symbolize a potential outcome based on the attribute evaluation. The tree's terminal nodes, known as leaf nodes, conclude the decision making process. The journey from the root node to any leaf node delineates a set of rules for classification. At every node, decisions are made based on whether a feature's value meets a certain criterion, typically formulated as $x_i \leq$ *threshold*, where x_i represents the attribute's value at node i, and the threshold is a predetermined constant derived during the model's training phase.

This study focuses on employing the Gini impurity as the primary metric for optimizing the decision-making process at each node. The Gini impurity is a measure designed to evaluate the frequency at which any randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Mathematically, it's defined as$\mathrm{Gini}(p) = 1 - \sum_{i=1}^{C} p_i^2$, where p_i denotes the proportion of elements belonging to class i within the node. This metric aids in selecting the most appropriate attribute and its corresponding threshold by minimizing the likelihood of incorrect classifications, thus steering the model training towards greater accuracy.

1.2.4 Support Vector Machines

Support Vector Machines (SVM) are highly regarded for their efficiency in classifying data with high dimensionality, which is pertinent to our scenario. The kernel function for Linear SVMs is defined as $K(x_i, x_j) = x_i^T x_j$, indicating a straightforward linear relationship between the input vectors. Despite the inherently binary nature of the classification problem addressed in the study, an one-vs-rest strategy was chosen for its implementation. In a scenario with *n* training samples each represented in *p* dimensions (where *n* Is the number of samples and *p* is the number of features per sample), SVMs aim to identify the hyperplane that not only separates the datapoints into two distinct classes but does so with the maximum margin. Essentially, this type of hyperplane divides the datapoints labeled as $y_i = 1$ from those labeled as $y_i = -1$, ensuring the greatest possible distance to the closest data point from either category.

Mathematically , the equation of any hyperplane can be expressed through the set of points x that fulfill the equation $w^T x - b = 0$, where *w* is a weight vector perpendicular to the hyperplane, and *b* is the bias term that provides the displacement of the hyperplane from the origin. The objective is to optimize *w* and *b* such at the margin, or the distance between the hyperplane and the nearest points from both classes, is maximized, thereby enhancing the classifier's discriminative power.

1.3 Oversampling/Undersampling methods

Oversampling and undersampling are strategies designed to address imbalances in the class distribution of a dataset. Their aim is to create a more balanced class distribution, thereby enhancing the performance of machine learning models that may struggle with skewed datasets.

Oversampling techniques work by increasing the number of instances in the minority class(es). This can be achieved through various methods, such as simply duplicating existing minority class samples or by generating new samples that are similar to the existing ones using algorithms like Synthetic Minority Oversampling Technique (SMOTE) and adaptive synthetic sampling (adasyn). The goal is to elevate the minority class to a level where it holds a more comparable representation relative to the majority class, thus reducing bias towards the majority class.

Conversely, undersampling involves reducing the number of instances in the majority class(es). This could mean randomly removing samples from the majority class to decrease its size and bring the class distribution closer to equilibrium (Near Miss). While effective in balancing the dataset, this approach risks losing potentially valuable information contained within the majority class samples that are discarded.

1.3.1 SMOTE

SMOTE, or Synthetic Minority Oversampling Technique, is a method used to generate new samples in the feature space to address the problem of class imbalance in machine learning datasets. It operates within a dataset that comprises n samples, each with m features.

The process begins by selecting a random sample from the minority class. For this chosen sample, SMOTE identifies its k nearest neighbors in the feature space; typically, $k = 5$ is chosen. The creation of a synthetic data point then follows this initial step.

To generates this new datapoint, SMOTE selects one of the k neighbors and constructs a vector from the chosen neighbor to the original sample. This vector is then multiplied by a random number x, which ranges between 0 and 1, effectively determining how far along the vector the synthetic point will be placed. Adding this scaled vector to the original sample point results in the creation of a new, synthetic data point within the feature space, thereby augmenting the minority class with artificially generated, yet plausible , samples. This technique helps in balancing the class distribution, enhancing the performance of machine learning models on imbalanced datasets.

1.3.2 Adaptive synthetic sampling

“The main idea is to use a weighted distribution for different groups within the minority class based on their level of learning difficulty. The more challenging the data is to learn, the higher the number of synthetic data generated. This approach enhances learning compared to data distributions in two ways: it reduces bias introduced by class imbalance and adaptively shifts the classification boundary with respect to data with insufficient representation.” (Méthode de rééquilibrage des classes en classification supervisée, Merwan CHELOUAH). The ADASYN algorithm, a variation of SMOTE, takes into account variance between points instead of linear correlations. It uses the density of observations, generating more data in areas where the density of sub-effects is lower. This density is calculated based on the number of observations from the majority class around the point where new synthetic samples are generated.

1.3.3 Near-Miss

The Near-Miss algorithm is an undersampling technique designed to help balance datasets by reducing the size of the majority class. It employs the k-nearest neighbors method to measure the similarity between data points, specifically aiming to identify and eliminate those majority class samples that are closest to the minority class samples.

In the process described, for each majority class sample, the algorithm finds it’s k-nearst neighbors wihtihn the minority class, with k typically set in this paper. It then calculates the aggregate distance from the majority class sample to these k-nearest minority class neighbors. The key idea is to prioritize the removal of majority class samples that are nearest (i.e., most similar) to the minority class, based on the rationale that these are less critical for defining the boundary between the classes. By removing the samples with the smallest total distance to the minority class, the algorithm effectively retains those majority class samples that are most dissimilar from the minority class, aiming to preserve the integrity of the class boundary and facilitate better classification performance.

It's important to note that there are three versions of the Near-Miss algorithm, each with its own approach to selecting which majority class samples to remove. The version detailed here is known as Near-Miss Version 1. This version specifically focuses on minimizing the presence of majority class samples that are closely resembling the minority class, thereby attempting to simplify the task of distinguishing between the two classes for a classification algorithm.

1.4 Receiver Operating Characteristic and Curve Score

The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) score are crucial metrics for evaluating the performance of models in binary classification problems. The ROC curve is a graphical representation that plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. It demonstrates the trade-off between sensitivity (or TPR) and specificity (1 - FPR) across different thresholds without committing to a particular classification cut-off.

A model that perfectly predicts all positives and negatives correctly would have an ROC curve that rises vertically up the y-axis and then moves horizontally along the x-axis, essentially forming a right angle and indicating perfect performance. Conversely, a model with no discriminative power, equivalent to random guessing, would produce a diagonal line from the bottom left corner to the top right corner of the ROC space.

The AUC score quantifies the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). This score provides a single scalar value that summarizes the model's performance across all threshold values, making it easier to compare different models. A higher AUC value indicates better model performance, with a score of 1.0 representing a perfect model and a score of 0.5 indicating a model that performs no better than random chance. The AUC score is particularly useful because it is independent of the classification threshold and provides an aggregated measure of performance across all possible classification thresholds.

2. Data (dataset Here)

This project dives into a dataset found on Kaggle,. It is a CSV file with 18 kinds of information. Our focus is mainly oriented on the job description if they're marked as fraudulent (Figure 1), if they have a logo (Figure 2), if experience is required (Figure 3), the employment type (Figure 4) and the required education (Figure 5). These precise data give us a lot to work with for spotting fake job ads.

One big challenge we notices is that the dataset is clearly imbalanced with more non-fraudulent than fraudulent job ads (more than ten times more). With a narrow look, we found that most of the fake ads were for full-time, entry level positions. This detail becomes clearer when looking at certain graphs in our report. After this we took a look at the most common words in both types of ads using the bag-of-words method (Figure 5), to see if we could guess a result based on this analysis

To better understand and work with the dataset, we cleaned up the text by lemmatizing: which simplifies words to their base format, and removing stop words: which don't add much meaning in the classification. We did this cleanup separately for the real and fake job ads in order to have a point of view well-marked on both side.

Below we have a realistic look on how the data is distributed between fraudulent and non-fraudulent interesting values.

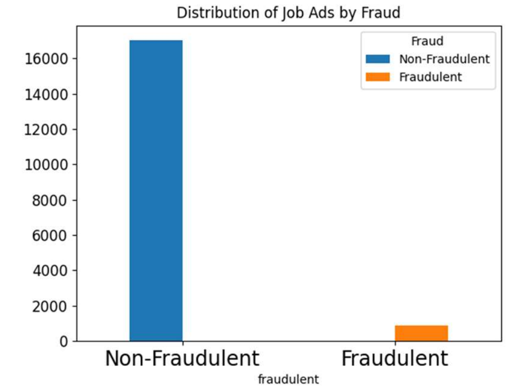

Figure 1 : Data's distribution

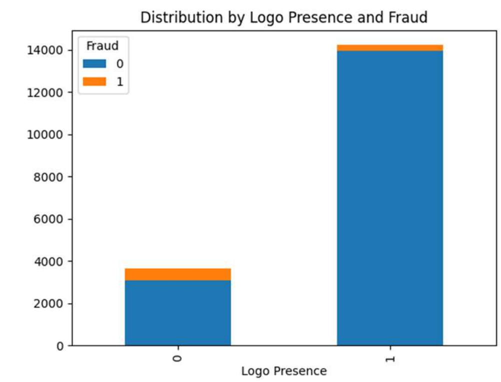

Figure 2: Logo Presence

Figure 3: Employment type

Figure 4: Experience Required

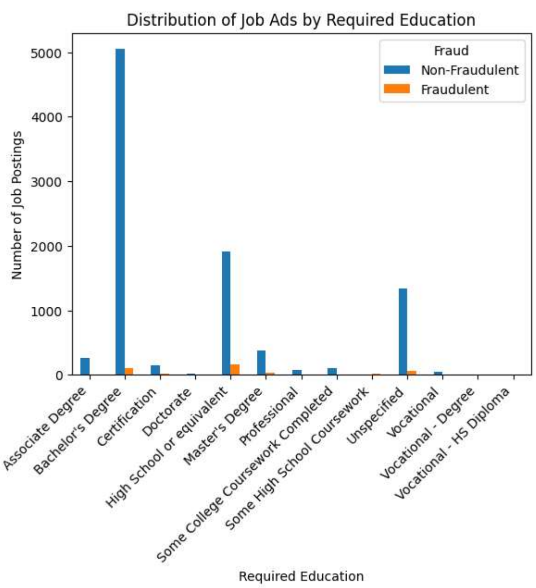

Figure 5:Required education

work	1167	team	19671
service	763	work	19584
customer	680	customer	14367
project	607	service	12002
amp	596	business	11109
product	591	company	11083
position	578	client	11002
team	552	product	10993
company	550	experience	10353
time	504	new	9474

Figure 5: Bag of 10 words for Fraudulent and non-Fraudulent job ads

3. Pipeline

3.1 Training count vectorizer

Transforming text into a format understandable by ML algorithm is crucial for processing and analysis. In our study, this transformation is achieved by employing a vectorizer on the training data. It's important to use the same vectorizer for both training and testing datasets to maintain the integrity of the evaluation process. If we were to fit the vectorizer on the testing data as well, it would inadvertently incorporate information from the test set into the training process, thereby contaminating the evaluation making it biased.

The rationale behind using the same vectorizer for both datasets is grounded in the assumption that the textual content across the two groups (fraudulent, non-fraudulent) is sufficiently similar. This similarity ensures that the vocabulary captured by the vectorizer from the training data is comprehensive enough to encompass all tokens found in the test data. By using this approach, we can accurately transform the test data into a machine readable format without leaking information between the datasets, thus preserving the validity of our evaluation metrics and conclusions drawn from the model performance.

3.2 Baselines

Establishing baselines is essential here as they act as reference points to assess the performance of advanced models. Theses baseline models set the initial benchmark for comparison, offering insight into how much improvement newer, more complex models of sampling provide.

In our study, we've defined four baseline models. Theses baselines are created without the application of any sampling techniques to address class imbalance. The intention with this approach is to get the raw performance of each algorithm in its most fundamental form, providing a clear measure of their effectiveness before any data manipulation strategies are employed.

This straightforward comparison allows us to identify the potential benefits and limitations of each algorithm as a starting point. This is the basis of our approach Thus, we can explore how different techniques might enhance the models' ability to handle imbalanced dataset.

3.3 Data manipulation strategies : Oversampling/Undersampling

In this step of our approach, we apply oversampling and undersampling techniques to the training dataset. Following this adjustment for class balance, we retrain our models using the same ML algorithms and maintain the hyperparameters established during the baseline phase for consistency. This process facilitates a direct comparison between the outcomes before and after the application of class balancing technics.

For this part of the study, the analysis was conducted under two distinct method: with probability (probabilistic approach) and without probability (non-probabilistic approach). The probabilistic approach evaluates the models' predictions in terms of likelihood, allowing us to measure not just the prediction itself but the model's certainty in making that prediction. The non-probabilistic approach, on the other hand, focuses on binary outcomes of the predictions without assessing the confidence level behind each prediction.

3.4 Probabilistic view

Last and not least A probabilistic approach allows us to go beyond a simple binary outcome, where each prediction is either one class or the other. Instead, the probabilistic view indicates the likelihood of an instance belonging to a particular class. This score offers a richer, more informative picture, as it encapsulates the model's certainty or uncertainty about its predictions. In practice, it enables us to: Asses confidence level, Examine stability, Identify tresholds for decision making, analyze trends, etc…

4. Results

Upon examining the performance metrics across various machine learning models, a notable observation emerges with the Naive Bayes algorithm. Initially, Naive Bayes presented the lowest AUC scores among the evaluated models (Baesline). However, the implementation of class balancing techniques such as SMOTE, Near-Miss, and ADASYN markedly enhanced its performance, with particularly significant improvements observed following the SMOTE and ADASYN in the three applications on Naive Bayes Logistic Regression and Decision Trees. This results is not really surprising since ADASYN is partially based on SMOTE algorithm.

	Technique	Modèle	Type	Valeur
0	SMOTE	Naive Bayes	Normal AUC	0.74
1	SMOTE	Naive Bayes	Proba AUC	0.78
2	SMOTE	Logistic Regression	Normal AUC	0.83
3	SMOTE	Logistic Regression	Proba AUC	0.87
4	SMOTE	Decision Trees	Normal AUC	0.83
5	SMOTE	SVM	Proba AUC	0.91
6	Near Miss	Naive Bayes	Normal AUC	0.53
7	Near Miss	Logistic Regression	Proba AUC	0.51
8	Near Miss	Decision Trees	Normal AUC	0.57
9	Near Miss	Naive Bayes	Proba AUC	0.62
10	Near Miss	Logistic Regression	Normal AUC	0.58
11	Near Miss	SVM	Proba AUC	0.59
12	ADASYN	Naive Bayes	Normal AUC	0.68
13	ADASYN	Naive Bayes	Proba AUC	0.71
14	ADASYN	Logistic Regression	Normal AUC	0.83
15	ADASYN	Logistic Regression	Proba AUC	0.87
16	ADASYN	Decision Trees	Normal AUC	0.78
17	ADASYN	Decision Trees	Proba AUC	0.87
18	ADASYN	SVM	Normal AUC	0.80
19	ADASYN	SVM	Proba AUC	NaN

Figure 6 : AUC Score

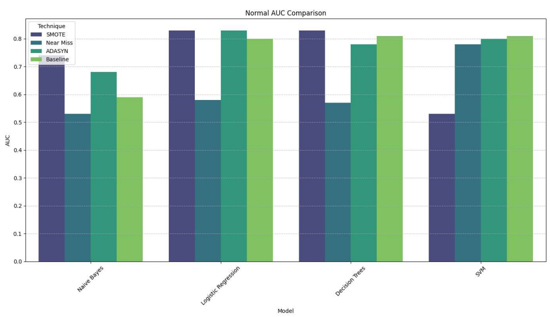

Figure 7 : Comparison Normal Score AUC

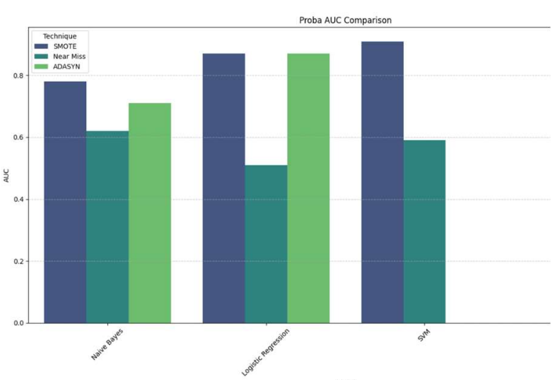

Figure 8 : Comparison Probabilistic Score AUC

Conversely, the performance metrics for other algorithms remained relatively stable across the baseline, SMOTE, and ADASYN interventions. These models demonstrated robust initial performances, with AUC scores consistently near 0.80, experiencing only slight improvements with SMOTE except with SVM model. When using SMOTE, the synthetic samples created for the minority class could be introducing noise or they might not represent the true underlying distribution accurately, leading to a more complex decision boundary that an SVM struggles to learn. This is where ADASYN is better. However, we notice a remarkable decrease for Near-Miss algorithm. Near Miss focuses on the points closest to the decision boundary, which might result in a biased sample that does not represent the true distribution of the majority class. Such significant information loss can adversely affect the model's performance, especially in complex models that rely on capturing intricate patterns in the data which is our case here (**Figure 5**)

Incorporating a probabilistic perspective into our analysis provides additional depth to our understanding of model performances. This approach allows us to evaluate not only the accuracy of the models' predictions but also the confidence levels associated with these predictions. The significant response of the Naive Bayes algorithm to data balancing techniques, as observed through probabilistic measures, underscores the importance of considering model certainty. This probabilistic analysis offers valuable insights into the dynamics between model performance and data preparation techniques, highlighting the nuanced benefits of employing probabilities in the evaluation of machine learning algorithms.

According to **Figure 8** reflecting on the Proba AUC values, we observe a compelling trend where models applying probabilistic methods demonstrate superior performance across all techniques, compared to their Normal AUC counterparts. This phenomenon is particularly notable with the ADASYN technique, where we see enhancements in the AUC scores for all models. For instance, the Naive Bayes model exhibits a climb from a Normal AUC of 0.68 to a Proba AUC of 0.71, and the Logistic Regression model ascends from 0.83 to 0.87 when employing probabilistic scoring. The Decision Trees model follows suit with an increase from 0.78 to 0.87. However, it's important to note that the Proba AUC value for the SVM model is not available (due to computing power), hence we cannot comment on the probabilistic improvement for this particular model under the ADASYN technique.

These findings underline the value of using probabilistic models, especially when dealing with imbalanced datasets treated with oversampling methods such as ADASYN, emphasizing the potential of probabilistic approaches in enhancing the model's ability to differentiate between classes more effectively. In addition, it helps to put aside other method like Near-Miss coupled with SVM algorithm : The AUC score is relatively low and the likelihood is not very consistent which provide a low confidence.

5. Discussion

In our exploration, we delved into the effects of class balancing techniques, such as oversampling and undersampling, on the efficacy of machine learning algorithms in identifying fraudulent job advertisements. Our research unveiled that, generally, the influence of these techniques on the algorithms tested did not markedly alter their performance, with the notable exception of the Naive Bayes classifier. This anomaly with Naive Bayes suggests an area ripe for further investigation in upcoming studies.

A significant addition to our study was the integration of a probabilistic approach to our analysis. This method allowed us to not only predict classifications but also to understand the confidence levels behind these predictions. Employing probabilistic view offers a deeper layer of insight, providing clarity on how certain the model is about its predictions, especially in complex scenarios like fraudulent job ad detection. This aspect of our research underscores the potential of probabilistic analysis in enhancing model interpretability and reliability. The probabilistic approach offers a nuanced understanding of model certainty, which could be instrumental in domains where the cost of misclassification is high. Consequently, the findings reinforce the argument for incorporating probabilistic measures into the evaluation metrics of machine learning models to achieve a more refined and dependable analysis.

Nonetheless, our investigation comes with its set of constraints. A notable limitation was our reliance on default hyperparameters as specified by the utilized Python libraries, without engaging in auto-tuning. This approach may have capped our ability to achieve and explore the full potential of the algorithms and the balancing techniques. Future studies could greatly benefit from auto-tuning across all algorithms, potentially unveiling optimized performance nuances.

Furthermore, our study could be expanded by incorporating a wider array of machine learning models, particularly those adept at handling both textual and numerical data. Given that our dataset encompasses numerous numerical attributes, such as required experience levels, leveraging models that can process this multifaceted data might yield more nuanced insights. Additionally, advancing the preprocessing phase could further refine the data quality, potentially leading to more accurate classification outcomes.

6. Conclusion and future work

A pivotal aspect of text mining involves preprocessing raw text to make it understandable for machine learning algorithms. This process entailed removing stopwords, and any non-alphabetical characters. Subsequently, each word was lemmatized, meaning it was converted to its base or dictionary form. This foundational step is crucial for clearing the noise in the data and ensuring that the algorithms can focus on the meaningful content within the text.

When we applied traditional machine learning methods without hyperparameter tuning to our data, we observed that the performance, as measured by the AUC score, remained relatively unchanged across different class balancing techniques, with the exception of the Naive Bayes classifier. This classifier showed a notable improvement, particularly with the use of SMOTE and ADASYN techniques, as evidenced in the results table **(Figure 6)** This distinct behavior of the Naive Bayes classifier compared to other algorithms raises intriguing questions for future research. Understanding why Naive Bayes benefits more significantly from class imbalance handling could provide valuable insights into its mechanics and potential optimizations.

These findings highlight the importance of selecting appropriate preprocessing techniques tailored to the specific dataset and problem domain. Moreover, the integration of probabilistic analysis, emphasizing its utility in enhancing the robustness and reliability of binary classification tasks, especially in complex and ambiguous contexts like fraudulent job ad detection. Therefore, incorporating both class balancing techniques and probabilistic measures can significantly enhance the effectiveness of machine learning models, ultimately contributing to more accurate and dependable analyses.

Moreover, a substantial challenge in this field is the acquisition of a robust dataset of fraudulent job ads. The scarcity of verified fraudulent ads complicates the task of building a comprehensive and reliable dataset. The uncertainty surrounding the authenticity of collected ads adds another layer of difficulty, highlighting the need for meticulous verification processes and the exploration of novel data collection strategies. This challenge underscores the ongoing struggle in leveraging machine learning for fraud detection, emphasizing the importance of quality data in developing effective and accurate predictive models.

Acknowledgments

I would like to express my profound appreciation to Professor Gerald Forhan for guiding me towards this research topic. His direction was particularly invaluable at a time when I was struggling to find substantial depth in my previous subject. Professor Forhan's insights and guidance not only clarified my path but also made it possible for me to delve deeply into an area that perfectly aligns with my interests.

References

Manning, Raghavan, and Schütze (2008), *Introduction to information retrieval.*

ChengXiang Zhai and Sean Massung: *Text Data Management and Analysis.*

Morgan & Claypool, 2016. *A Practical Introduction to Information Retrieval and Text Mining.*

Barocas et al. (2016), *Big Data's Disparate Impact*

Feng Li and al. (2020) *A Novel Simplified Convolutional Neural Network Classification Algorithm of Motor Imagery EEG Signals Based on Deep Learning*

Merwan Chelouah, *MÉTHODES DE RÉÉQUILIBRAGE DES CLASSES en classification supervisée*

Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn *: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*

Marcel Bollman, *TDDE16*

Sokratis Vidros, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu. 2017. Au tomatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. Future Internet, 9(1).

Wikipedia : Bag of words, Baseline, Bernoulli Naïve Bayes, Decision Trees, Support vector machine, Logistic Regression, Smote, Receiver operating characteristics