Machine Learning Project, 2023, 1–8

September Exam Session - Università degli Studi di Milano Bicocca, CdLM Data Science

Safeguarding Employment: Unmasking Fraudulent vs. Legitimate Jobs with Machine Learning

Matteo Pasotti

Abstract

This project aims to answer the research question of whether it is possible to differentiate between legitimate and fraudulent job advertisements using the conventional information available on job search websites. The dataset utilized contains a range of information, but it suffers from a significant class imbalance, with a shortage of fraudulent job ads. To tackle this challenge, we thoroughly analyzed and processed the dataset's features to render them suitable for applying Machine Learning techniques. We conducted multiple classification methods and, following a rigorous comparison of these approaches, we identified an optimal solution. Our objective was to discover a solution that provided satisfactory outcomes in terms of both model performance and the reliability of the analysis itself.

Key words: Jobs Announcement; Machine Learning; Imbalanced Class

Contents

INTRODUCTION

Scams associated with fake job advertisements present an ongoing menace, especially for individuals actively pursuing professional opportunities. These scams have become increasingly sophisticated, often managing to deceive even cautious individuals due to their ability to seamlessly blend in with the multitude of legitimate job postings. There are several telltale signs that should raise suspicions of a potential scam. These indicators include job offers that appear excessively lucrative, demands for sensitive personal information at an early stage in the hiring process, and the rapid offering of a job without the customary interviews or onsite visits.

Employment-related scams inflict substantial financial losses upon workers, with the Better Business Bureau reporting an estimated $2 billion in direct losses annually. In just the first quarter of 2022, as many as 14 million individuals found themselves exposed to employment-related scams. The Federal Trade Commission further revealed that unwitting job seekers suffered financial losses totaling $68 million due to counterfeit job opportunities. These fraudulent activities encompass fake job advertisements, work-from-home scams, deceptive interview practices, and government-related fraud schemes.Furthermore, there is a troubling surge in employment-related scams across various websites, including

Compiled on: September 3, 2023.
Draft manuscript prepared by the author.

platforms like Facebook, LinkedIn, and Indeed. These job postings may appear alluring, offering the promise of high wages, but they often demand upfront personal information, including social security numbers. However, lurking behind these seemingly enticing ads are frequently cybercriminals intent on stealing individuals' identities, which they subsequently employ to perpetrate various forms of fraud, such as making fraudulent claims for unemployment benefits. This rapidly proliferating scam has adversely impacted millions of job seekers.

It is of utmost importance for job seekers to exercise extreme caution and diligence when engaging with online job opportunities. It is crucial to meticulously verify the legitimacy of job postings, refrain from divulging sensitive personal information to unfamiliar parties, and always conduct due diligence on the company or organization before proceeding with any hiring process. Prevention serves as the primary means of safeguarding oneself from these scams, which can result in severe financial and privacy repercussions.
In this study, we utilized a dataset that contained information commonly found in job advertisements, excluding features that could easily distinguish a scam. The primary objective of this research was to develop machine learning models capable of predicting whether a job advertisement is authentic or fraudulent. Subsequently, we compared the results obtained to identify the best-performing model. The ultimate aim of this analysis is to furnish valuable insights for individuals in search of employment or improved job prospects.

Dataset Description

The dataset encompasses 17,880 job descriptions, of which roughly 800 are deemed fraudulent. It includes both textual information and meta-information pertaining to these job listings. Below, we provide a concise overview of the various features within the dataset:

- **job_id**: A unique identifier for each job listing.
- **title**: The job ad's title.
- **location**: The geographical location specified in the job ad.
- **department**: The corporate department associated with the job (e.g., sales).
- **salary_range**: An indicative salary range (e.g., $50,000–$60,000).
- **company_profile**: A brief description of the company.
- **description**: A detailed description of the job.
- **requirements**: The listed requirements for the job opening.
- **benefits**: The benefits offered by the employer.
- **#telecommuting**: True for telecommuting positions.
- **#has_company_logo**: True if company logo is present.
- **#has_questions**: True if screening questions are present.
- **employment_type**: The type of employment (e.g., Full-time, Part-time, Contract).
- **required_experience**: The required level of experience (e.g., Executive, Entry level, Intern).
- **required_education**: The required educational qualifications (e.g., Doctorate, Master's Degree, Bachelor).
- **industry**: The industry to which the job belongs (e.g., Automotive, IT, Health care, Real estate).
- **function**: The function or role of the job (e.g., Consulting, Engineering, Research, Sales).
- **#fraudulent**: The target attribute used for classification, with values indicating whether a job ad is fraudulent or not
  Note: The "#" symbol indicates binary features.

It's worth noting that this dataset exhibits significant class imbalance, with only 800 out of 18,000 job postings identified as fraudulent. Additionally, there is a substantial number of missing values across nearly all features, totaling 67,148 missing entries.

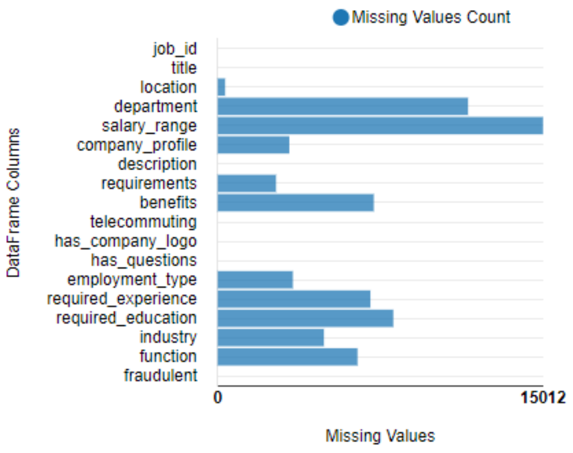


Figure 1. Missing Value foreach Feature

Data Preparation

The preprocessing phase played a pivotal role in preparing the dataset for model training and in-depth feature analysis. We initially examined the string variable "location," which housed information about the company's job offer location. This feature maintained a consistent structure across all rows, encompassing details such as country, state, and city (e.g., "US, NY, New York"). However, not all fields were consistently filled, resulting in some instances marked as 0-length strings and subsequently converted into Missing Values for later handling. To enhance data organization, we split the "location" field into three distinct variables: "Country," "State," and "City," using a comma as the delimiter.
Moving forward, our attention shifted to two other variables: "title" and "description," which contained the job posting's title and its accompanying description. Remarkably, these two features exhibited no missing values. Our approach to adapt these variables involved calculating their respective lengths, recognizing the potential significance of length as a crucial factor in determining the authenticity of a job posting.
Upon inspecting the "description" column, we observed that many companies included additional information such as URLs, phone numbers, and email addresses. We created three new columns to capture this supplementary data by employing regular expressions. In cases where these details were absent, the corresponding fields were duly marked as "null."

Handling Missing Values

As mentioned earlier, the dataset contains a high number of missing values, but all the features with missing values are string-based. Managing these missing values was a relatively straightforward process. We replaced every null value in the dataset with the string "Unspecified," indicating the absence of that particular data point.
Some features had a limited and well-defined set of possible values, such as "employment_type," while others had a vast number of possible values. The idea for these variables was to use a regular expression to indicate the presence or absence of data, assigning a value of 0 if the field was "Unspecified" and 1 if specified. This simplification of the last data I mentioned has made it more suitable for analysis, as these features originally contained numerous and scattered text values.1 For the data with limited values, we applied a Label Encoding technique. In this method, each category was assigned to a unique integer, and even the term "Unspecified" was treated as a category. This approach is particularly useful when there is an inherent order among the categories.

Correlation

Correlation is a measure that assesses how two variables are related to each other. It can be positive (an increase in one variable corresponds to an increase in the other), negative (an increase in one variable corresponds to a decrease in the other), or neutral (no evident relationship). Correlation helps understand how variables mutually influence each other. In our case, all features were used except for "job_id," "title," and "description," with the latter two being replaced by their respective lengths.

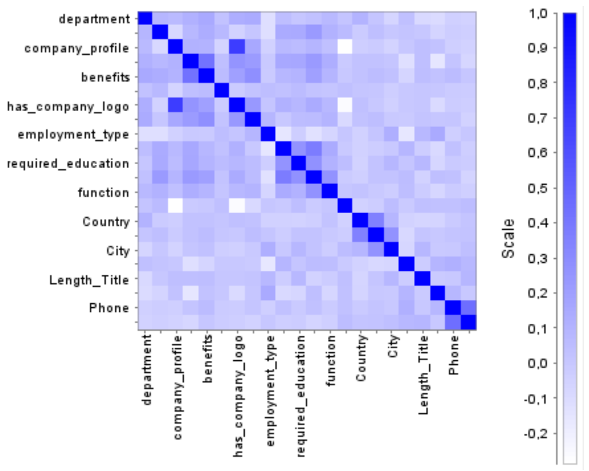


Figure 2. Correlation between Features

The heatmap reveals varying degrees of correlation represented by different color gradients. One particularly noteworthy correlation is observed between "company_profile" and "has_company_logo," which stands at a significant 0.694. This is followed by the correlation between "Phone" and "Email" at 0.449, and "requirements" and "benefits" at 0.426. The remaining features, upon analysis, do not display substantial correlations. However, when we consider only correlations greater than 0,500 as meaningful, the first correlation proves to be the most significant. This led us to the decision to exclude the variable "has_company_logo" from the analysis. This choice was made because it exhibited the lowest correlation with the target variable "fraudulent" in comparison to "company_profile."

VIF

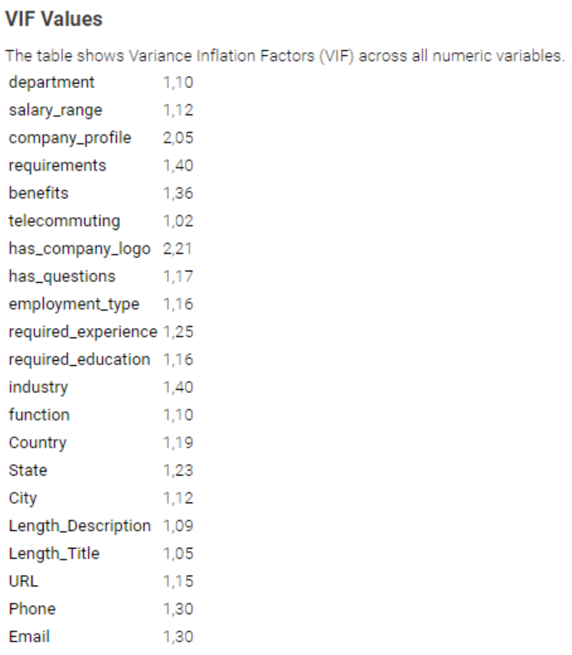
The Variance Inflation Factor (VIF) is a metric used to assess multicollinearity among predictor variables in a regression model. Multicollinearity occurs when two or more predictor variables are strongly correlated with each other, which can lead to issues in interpreting model coefficients and the stability of predictions. The VIF measures how much the accuracy of regression coefficient estimates is influenced by multicollinearity.

The VIF for a specific predictor variable is calculated as:

$$VIF_i = \frac{1}{1 - R_i^2}$$

Here R_i^2 is the coefficient of determination of a linear regression in which the variable represents the coefficient of determination of a linear regression in which the variable i is predicted using all the other predictor variables. In general, a high VIF indicates significant multicollinearity, suggesting that the variable under consideration exhibits a strong correlation with other variables within the model.

In general, a high VIF indicates strong multicollinearity and suggests that the variable in question is highly correlated with other variables in the model.



VIF Values

The table shows Variance Inflation Factors (VIF) across all numeric variables.

department	1,10
salary_range	1,12
company_profile	2,05
requirements	1,40
benefits	1,36
telecommuting	1,02
has_company_logo	2,21
has_questions	1,17
employment_type	1,16
required_experience	1,25
required_education	1,16
industry	1,40
function	1,10
Country	1,19
State	1,23
City	1,12
Length_Description	1,09
Length_Title	1,05
URL	1,15
Phone	1,30
Email	1,30

Figure 3. VIF

The threshold we consider acceptable, indicating that it does not represent severe multicollinearity, is less than 5. This threshold enables us to incorporate all the variables in the dataset into the model without encountering significant issues.

Models

In this project, we applied various Machine Learning classification techniques to assess the accuracy of job advertisements, aiming to determine the most suitable technique based on the available data. We employed the following methods:

- **Random Sampling Only**: Data was partitioned randomly both during the initial partitioning and in cross-validation.
- **Stratified Sampling Only**: We used stratified partitioning both initially and in cross-validation.
- **SMOTE (Synthetic Minority Over-sampling Technique)**: To address the minority class imbalance, we applied SMOTE to oversample it.
- **Bootstrapping**: To handle class imbalance, we utilized Bootstrap Sampling to oversample the minority class and undersampled the majority class.
- **Undersampling**: We performed undersampling on the majority class
- **Undersampling - Oversampling**: Combining SMOTE for oversampling the minority class with undersampling the majority class.

XGBoost

The model employed for all the techniques was XGBoost. XGBoost Tree Ensemble is an exceptionally effective machine learning model. It is built upon the concept of constructing an ensemble of decision trees, referred to as weak trees, where each new tree aims to rectify the errors of the previous ones. This iterative process consistently enhances the model's performance. XGBoost offers numerous advantageous features, such as intelligent handling of missing data, the ability to fine-tune the model to prevent overfitting, and flexibility in using custom objective functions. One of its major strengths lies in its incredible speed and scalability. It can be utilized with large datasets and leverages parallelism on both CPU and GPU to train models rapidly.

Parameter Optimization

Parameter optimization in XGBoost Tree Ensemble has been a pivotal aspect of the model development process to achieve peak performance. XGBoost offers an array of parameters that can be fine-tuned to tailor the model to the data and the specific problem at hand. Some of the key parameters optimized include:

- **max_depth**: This parameter governs the maximum depth of decision trees within the ensemble. Adjusting the value of `max_depth` allows for the regulation of tree complexity, thus controlling overfitting or underfitting.
- **subsample**: It sets the fraction of training data to be randomly sampled during the construction of each tree. This parameter can influence the model's variance and aid in mitigating overfitting.
- **eta**: Also known as the learning rate, it controls the speed at which the model learns from the data. Reducing `eta` slows down learning, which can be beneficial to prevent oscillations during convergence.
- **gamma**: This parameter oversees regularization, and a higher `gamma` value makes the model more conservative, thereby helping to preempt overfitting.
- **alpha** and **lambda**: These parameters are employed to apply L1 (lasso) and L2 (ridge) regularization to the model, respectively. By adjusting `alpha` and `lambda`, one can regulate the model's complexity and reduce overfitting.

X-Cross Validation

Cross-Validation is a fundamental technique in the field of machine learning, used to assess a model's performance robustly and reliably. It involves dividing the dataset into multiple parts, iteratively training and testing the model on these portions, and calculating evaluation metrics to obtain an accurate estimate of the model's performance.In this technique, the dataset is divided into "k" equal subsets known as "folds" (in our case, 10 folds). The model is then trained and tested "k" times, using each fold as a test set once while the other folds serve as the training set.Cross-Validation is essential because it helps prevent overfitting and underfitting of the model, allowing us to evaluate how the model performs on data it has never encountered during training. Additionally, it provides an estimate of the model's performance that is less influenced by the specific data distribution in the training set.

Imbalanced Dataset

As mentioned earlier, the dataset exhibited a significant class imbalance. We employed strategies such as XGBoost, X-Cross Validation, and Parameter Optimization collectively to tackle this issue because:

- **XGBoost**: This method provides specific parameters to handle class imbalance effectively.
- **X-Cross Validation**: It employs stratified sampling to ensure an equitable distribution of classes in each fold, facilitating a fair evaluation of the model.
- **Parameter Optimization**: This technique fine-tunes algorithm parameters (such as those in XGBoost) to maximize performance on imbalanced data, utilizing approaches like grid search or Bayesian optimization.

Accuracy

However, it has been observed that when using classification models, both parametric and non-parametric, with imbalanced datasets, they tend to exhibit high accuracy for frequent events and low accuracy for rare events. The accuracy is calculated using the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP and TN represent the number of instances correctly classified as belonging to the positive and negative classes, respectively.
- FP and FN indicate the number of instances (negative and positive) that are classified incorrectly.

Therefore, it becomes necessary to rely on other evaluation measures. Even with the utilization of the strategies mentioned above, the model tends to prioritize the frequent class, often overlooking the rare class.

Additional solutions for imbalanced datasets

In the techniques employed in our models, we utilized additional methods to effectively address the Imbalanced Class problem. Central to this effort was the principle of Oversampling and Undersampling, which were employed either individually or in combination:

- **Undersampling**: Undersampling involves reducing the size of the dataset of the dominant class (the one with more samples) to match the size of the minority class (the one with fewer samples). This entails randomly removing some samples from the dominant class. The objective is to balance the class proportions in the dataset, but it may result in the loss of important information, especially if the original dataset is already small.
- **Oversampling**: Oversampling is a technique wherein we increase the size of the dataset of the minority class to match the size of the dominant class. This can be achieved by replicating existing samples or generating new artificial samples. The goal of oversampling is to retain more information from the minority class without reducing the size of the dominant class. However, excessive oversampling can lead to overfitting.

These two techniques were applied using SMOTE and Bootstrap Sampling:

- **SMOTE (Synthetic Minority Over-sampling Technique)**: SMOTE is an oversampling technique that, instead of replicating existing samples from the minority class, creates synthetic samples by 'imagining' new examples that are a combination of existing samples close to each other in the dataset. This helps increase the number of minority class samples without simply duplicating them, reducing the risk of overfitting.
- **Bootstrap Sampling**: Bootstrap sampling is a technique used to create training samples randomly and with replacement. In other words, samples are randomly selected from the original dataset, and each selected sample is placed back into the original dataset before selecting the next one. This technique can be used to create multiple training subsets, each with some repetitions and omissions of original samples.

Performance evaluation

As previously explained, utilizing Accuracy as a metric to assess classifier performance is not the most suitable approach when dealing with imbalanced data. Therefore, we consider other criteria to provide a more comprehensive evaluation of performance. Specifically, we choose to calculate Recall, Precision and the F-measure.

Recall, also known as the True Positive Rate, is defined as follows:

$$\text{Recall} = \frac{TP}{TP + FN}$$

It represents the fraction of correctly classified positive records by the model. A high Recall value indicates a low percentage of misclassified positive records.

On the other hand, Precision describes the fraction of records that are genuinely positive among all those predicted as such:

$$\text{Precision} = \frac{TP}{TP + FP}$$

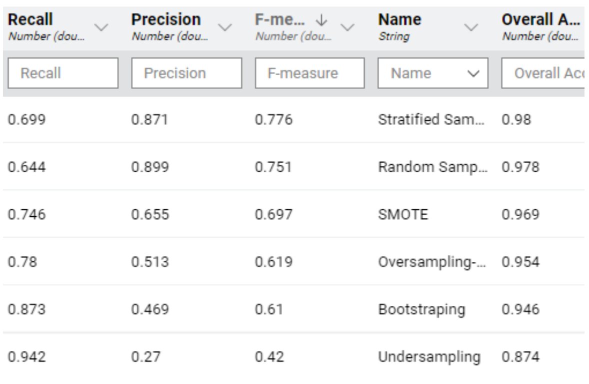
Higher Precision results in fewer false positives.

In some cases, these two measures may conflict: increasing the true positives of the rare class may improve Recall but simultaneously worsen Precision because it could lead to an increase in false positives. To mitigate this effect, we examine alternative measures that integrate these two evaluation criteria, such as the F-measure:

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Analysis and Results

During the initial analysis phase, the results were as follows:



Recall Number (dou...	Precision Number (dou...	F-me... Number (dou...	Name String	Overall A... Number (dou...
Recall	Precision	F-measure	Name	Overall Acc
0.699	0.871	0.776	Stratified Sam...	0.98
0.644	0.899	0.751	Random Samp...	0.978
0.746	0.655	0.697	SMOTE	0.969
0.78	0.513	0.619	Oversampling-...	0.954
0.873	0.469	0.61	Bootstraping	0.946
0.942	0.27	0.42	Undersampling	0.874

Figure 4. First Results

Let's analyze the performance of various techniques:

i. **Random Sampling**:

- High precision, indicating accuracy in predicting positives.
- Relatively low recall, potentially missing some positive cases.
- Moderate F-measure, striking a balance between precision and recall.
- Overall accuracy is relatively high

ii. **Stratified Sampling**:

- Shares characteristics with Random Sampling.
- Slightly higher recall and F-measure, possibly better at capturing positive cases.

iii. **SMOTE**:

- Relatively high recall, capturing more positives
- Lower precision, resulting in more false positives.
- Maintains a high overall accuracy.

iv. **Bootstrapping**:

- Very high recall, capturing most positive cases.
- Lower precision, leading to many false positives.

- Moderate F-measure and good, though slightly lower, overall accuracy.

v. **Undersampling**:

- Very high recall, capturing nearly all positive cases.
- Low precision, resulting in a low F-measure and overall accuracy.
- Tends to classify too many cases as positive.

vi. **Oversampling-Undersampling**:

- Strikes a balance between recall and precision.
- Results in a moderate F-measure and good overall accuracy.

If our primary goal is to prioritize precision, two suitable sampling techniques are Random Sampling and Stratified Sampling. If our aim is to maximize the capture of positive cases, Undersampling could be an effective choice. For a balance between recall and precision, SMOTE and Oversampling-Undersampling techniques come into play. On the other hand, Bootstrapping leans heavily towards emphasizing recall, albeit at the cost of precision.

SHAP

SHAP (SHapley Additive exPlanations) stands out as a machine learning model interpretation technique, offering the capability to dissect predictions and elucidate the role played by each variable. This approach facilitates a profound comprehension of why a model arrived at a particular prediction while also allowing for the assessment of variable importance. One of SHAP's strengths lies in its consistency, fairness, and versatility across a spectrum of models.By employing SHAP to compute Shap values for each model, I could discern which features held genuine significance in aiding the model's performance and which ones carried minimal weight in determining the target class.

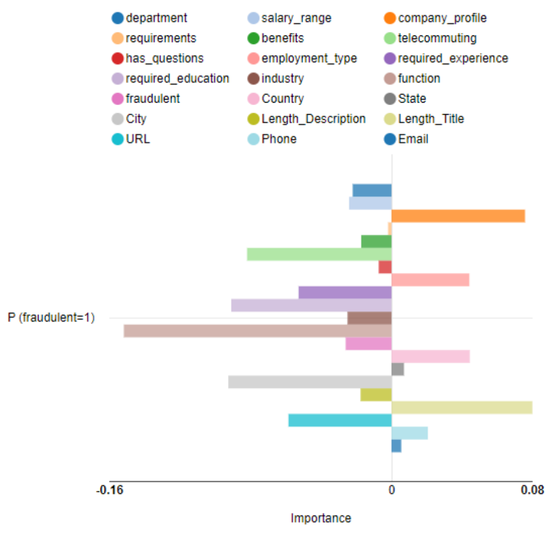
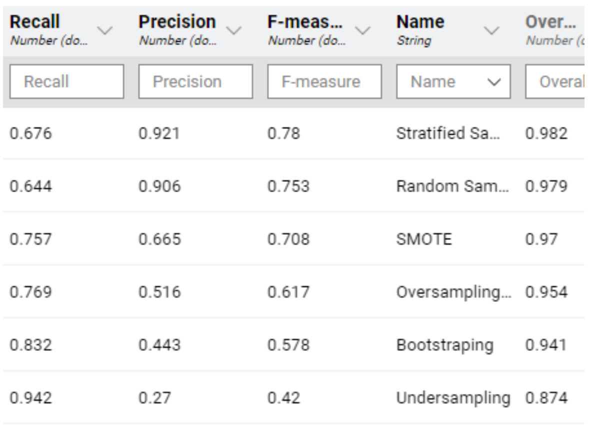


Figure 5. Shap Results for Random Sampling

I chose to retrain the model by excluding features that had minimal impact, and this decision resulted in a change in the outcomes.



Recall Number (do...	Precision Number (do...	F-meas... Number (do...	Name String	Over... Number (
Recall	Precision	F-measure	Name	Overal
0.676	0.921	0.78	Stratified Sa...	0.982
0.644	0.906	0.753	Random Sam...	0.979
0.757	0.665	0.708	SMOTE	0.97
0.769	0.516	0.617	Oversampling...	0.954
0.832	0.443	0.578	Bootstraping	0.941
0.942	0.27	0.42	Undersampling	0.874

Figure 6. Results with Shap

i. ***Random Sampling***:

- *Precision*: Increased from 0.89922 to 0.90625.
- *F-measure*: Improved from 0.7508 to 0.7532.
- *Overall Accuracy*: Slightly increased from 0.9785 to 0.9787.

ii. ***Stratified Sampling***:

- *Precision*: Improved from 0.8705 to 0.92126.
- *F-measure*: Increased from 0.7756 to 0.78.
- *Overall Accuracy*: Enhanced from 0.9804 to 0.9815.

iii. ***SMOTE***:

- *Precision* and *F-measure*: Remained similar.
- *Overall Accuracy*: Slightly increased from 0.9687 to 0.9698.

iv. ***Bootstrapping***:

- *Precision*: Increased from 0.46894 to 0.44308.
- *F-measure*: Improved from 0.6101 to 0.5783.
- *Overall Accuracy*: Decreased from 0.9460 to 0.9413.

v. ***Undersampling***:

- Results remained consistent between the two sets.

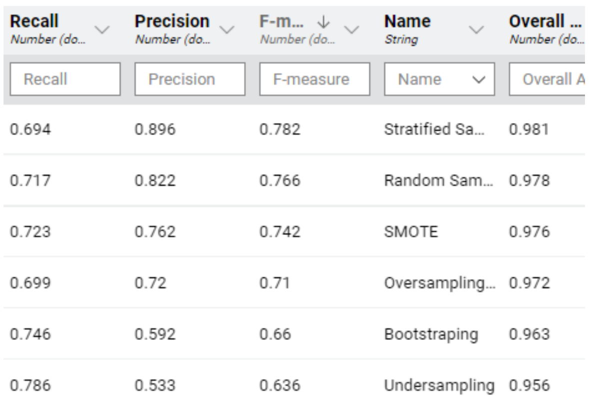
vi. ***Oversampling-Undersampling***:

- *Precision*: Increased from 0.5133 to 0.5155.
- *F-measure*: Improved from 0.6172 to 0.6193.
- *Overall Accuracy*: Slightly increased from 0.9536 to 0.9539.

Upon a comprehensive examination of these statistics, it becomes evident that different sampling techniques have distinct effects on model performance. Some techniques, notably Random Sampling and Stratified Sampling, have led to improvements in key metrics such as Precision, F-measure, and Overall Accuracy. On the other hand, Bootstraping exhibited a decrease in Precision and F-measure, resulting in a decline in Overall Accuracy. SMOTE and Oversampling-Undersampling showed stability in Precision and F-measure while achieving a slight increase in Overall Accuracy. Undersampling maintained consistency across both sets.

Optimizing F-Measure through Various Thresholds

The threshold serves as a pivotal value that dictates how the model categorizes its predictions as positive or negative. By adjusting the threshold, we sought to strike a balance between the model's precision and recall, ultimately maximizing the F-Measure. This approach enabled us to fine-tune the model's effectiveness within the specific context of our problem, considering the requirements and priorities of our application.



Recall Number (do...	Precision Number (do...	F-m... ↓ Number (do...	Name String	Overall ... Number (do...
Recall	Precision	F-measure	Name	Overall A
0.694	0.896	0.782	Stratified Sa...	0.981
0.717	0.822	0.766	Random Sam...	0.978
0.723	0.762	0.742	SMOTE	0.976
0.699	0.72	0.71	Oversampling...	0.972
0.746	0.592	0.66	Bootstraping	0.963
0.786	0.533	0.636	Undersampling	0.956

Figure 7. Shap and Threshold Results

i. **Random Sampling**:

- Recall increased from 0.644 to 0.717.
- Precision decreased from 0.906 to 0.822.
- F-measure increased from 0.753 to 0.766.
- Overall Accuracy slightly decreased from 0.979 to 0.978.

ii. **Stratified Sampling**:

- Recall increased from 0.676 to 0.694.
- Precision slightly decreased from 0.921 to 0.896.
- F-measure increased from 0.78 to 0.782.
- Overall Accuracy slightly decreased from 0.982 to 0.981.

iii. **SMOTE**:

- Recall decreased from 0.757 to 0.723.
- Precision increased from 0.665 to 0.762.
- F-measure increased from 0.708 to 0.742.
- Overall Accuracy increased from 0.97 to 0.976.

iv. **Bootstrapping**:

- Recall decreased from 0.832 to 0.746.
- Precision significantly increased from 0.443 to 0.592.
- F-measure increased from 0.578 to 0.660.
- Overall Accuracy increased from 0.941 to 0.963.

v. **Undersampling**:

- Recall significantly decreased from 0.942 to 0.786.
- Precision significantly increased from 0.27 to 0.533.
- F-measure significantly increased from 0.42 to 0.636.
- Overall Accuracy increased from 0.874 to 0.956.

vi. **Oversampling-Undersampling**:

- Recall slightly decreased from 0.769 to 0.699.
- Precision slightly increased from 0.516 to 0.720.
- F-measure increased from 0.617 to 0.709.
- Overall Accuracy increased from 0.954 to 0.972.

In summary, the overall performance across various methods shows an improvement in F-measure and overall accuracy. However, the changes in precision vary significantly, with some methods improving while others decrease. Recall also shows mixed changes, with some methods improving and others decreasing. Overall, the choice of the best method would depend on the specific goals and trade-offs in precision and recall that are acceptable for the task at hand.

Conclusion

In this study, our primary objective was to determine the most effective Machine Learning technique for distinguishing between fraudulent and legitimate job listings. Our ultimate aim is to enhance the job-seeking experience for candidates by reducing the risk of deception and fraud. To achieve this, we thoroughly utilized the available dataset, striving to extract valuable information to construct the optimal model.

- **Initial Analysis:** We began with a correlation analysis to streamline the dataset, eliminating one variable that was found to be redundant. Subsequently, we performed data preprocessing to prepare it for the training phase.
- **Addressing Imbalance:** We encountered a significant imbalance in the dataset, with fraudulent job listings being the minority class. To tackle this challenge, we applied various techniques across different models. These techniques included X-Cross Validation and Parameter Optimization, and we consistently employed the XGBoost algorithm due to its exceptional performance.
- **Data Balancing:** To further address the class imbalance, we utilized both Oversampling and Undersampling techniques. These approaches aimed to improve model performance by ensuring that the minority class (fraudulent job listings) was adequately represented in the training data.
- **Performance Metrics:** Our evaluation criteria primarily focused on Precision, Recall, and F-Measure for the minority class. Precision is crucial for minimizing false positives, ensuring that legitimate job listings are not misclassified as fraudulent, while Recall is vital for accurately identifying true fraudulent job listings.
- **Enhancing Model Performance:** In our pursuit of improving the model's performance, we employed additional techniques such as Shap and Threshold techniques. These were specifically geared towards enhancing the F-Measure.
- **Choosing Between Sensibility and Precision:** The choice between Sensibility (Recall) and Precision depends on the specific objectives. Precision minimizes false positives, crucial for protecting legitimate job listings from being wrongly labeled as fraudulent. Conversely, Recall is vital for correctly identifying true fraudulent job listings.
- **Potential Future Enhancements:** To further enrich our analysis, we have identified several potential improvements:
  - **Deep Learning:** Exploring Deep Learning techniques could involve counting the actual number of words within text-heavy features, such as job descriptions.

This approach may uncover words that have differing prevalence in fraudulent and legitimate job listings.

– **Advanced Analysis and ML Modeling:** Delving into advanced Machine Learning models, including BERT, Word2Vec, state-of-the-art Transformers, and more, holds the potential to further enhance model performance and metrics.
– **Web Application Deployment:** After finalizing the model, we envision the creation of a user-friendly web application using technologies like Flask or Streamlit. Deploying this application on the cloud would enable broader access and utilization of the model for future applications, benefiting a wider audience.

Bibliography

1. Kelly J. Fake Job Scams Are Becoming More Common—Here's How To Protect Yourself. Blog 2023 January;Https://www.forbes.com/sites/jackkelly/2023/06/01/fake-job-scams-are-becoming-more-common-heres-how-to-protect-yourself/.
2. Group O. What Is a Fake Job Posting and Ways to Spot it? Blog 2021;Https://www.omnesgroup.com/fake-job-posting/.
3. Bansal S. Real / Fake Job Posting Prediction. Dataset 2019;Https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction/.
4. Sharma S. Real / Fake Job Posting Prediction. BLOG 2023;Https://medium.com/@sohilsharma1996/real-fake-job-posting-prediction-e67eedfbccc4/.