BT4012 Report

Name	Matric No.
Wang Liang Bing	A0242199X
Yong Duan Kang	A0240563L
Soh Zuan, Azumi	A0239407B
Haris Bin Dzulkifli	A0235244L
Tay Zhi Yi	A0233288Y

Github Repo: https://github.com/dkcodes2/Fraudulent-Job-Postings-Detection

Contents Page

1 Problem Statement

Following the pandemic, coupled with the recent layoffs across multiple industries, job hunting has gotten exponentially harder. The motivation of this project is to protect the interest of job seekers. Fraudulent postings can be a means to collect personal and sensitive information from applicants, leading to identity theft. Additionally, fraudulent job postings could implicate job platforms and companies listed. Job boards and platforms rely on the trust of their users, which erodes with the presence of fake job postings, leading to decreased platform usage. A decline in trust can also lead to reduced revenue for job platforms that operate on a subscription basis or through advertising.

2 Dataset

We utilised the 'Real / Fake Job Posting Prediction' Kaggle dataset. This dataset comprises approximately 18,000 job listings of which about 800 are identified as fraudulent, hence having a class ratio of 4.70%, It contains 16 feature variables in total, 10 categorical, 1 numerical and 5 textual variables and meta-information about the jobs, serving as a resource for building classification models to detect fraudulent job postings. The features observed in this dataset can be found in our appendix (Figure 1).

2.1 Data Cleaning

2.1.1 Handing Location Data

In the `location` column, each entry contains multiple location-related details. For the purpose of our analysis, we will retain only the country code from each entry. This will involve splitting each location string by their commas and selecting the first segment, the country code, to ensure relevance in our dataset.

2.1.2 Handling Salary_Range Data

Since values in the `salary_range` column are formatted in a range, we will handle it by splitting the ranges into a lower and upper bound. This creates two new columns, `salary_lower_bound` and `salary_upper_bound`.

2.1.3 Handling Missing Data

The `salary_range` column is missing 15012 (84.0%) rows, and the `employment type` column is missing 3471 (19.4%) rows. `required_experience` and `required_education` are missing 7050 (39.4%) and 8105 (45.3%) rows respectively. Columns "company_profile", "description", "requirements", "benefits", "industry", "function", and "department", are text data columns which will be concatenated and tokenized further on. Thus, we will not handle missing values for these columns.

To address the missing values in the non-textual data columns, we will investigate each individual column.

2.1.3.1 Employment Type Data (Missing at Random)

In the rows where `employment_type` has the value 'Other', it can be observed from the title that the role actually takes on a range of more unique job types such as internships or trainees. On inspection of rows where

`employment_type` is 'Nan', the job title similarly states unique job types such as Traineeship and Assistant. Thus, the rows with 'Nan' values could be categorised as 'Other'.

2.1.3.2 Salary Data (Missing completely at Random)

We observe that the missing `salary_range` values are not obviously related to other columns and that there is no observable pattern to which they are missing. Our secondary research indicates that a large proportion of job listings across various sectors do not disclose salary information. This trend appears consistent regardless of job type, company, or industry. Given this widespread practice, the absence of salary data in our dataset seems to reflect a general norm rather than a factor influenced by specific characteristics of the job listings. As such, we assume they are missing completely at random (MCAR). We will use KNN imputation to fill in the missing columns. This will be done after the Train-Test split to prevent data leakage.

2.1.3.3 Required Experience Data

There exists the entry ‘Not Applicable’ in the `required_experience` column. There is no striking observable difference when compared to rows where it is ‘NaN’. Thus, the presence of ‘Nan’ values could suggest that they are ‘Not Applicable’.

2.1.3.4 Required Education Data

There exists the entry ‘Unspecified’ in the `required_education` column. There is no striking observable difference when compared to rows where it is ‘NaN’. Thus, the presence of ‘Nan’ values could suggest that they are ‘Unspecified’

2.2 Exploratory Data Analysis

2.2.1 Fraudulent Labels

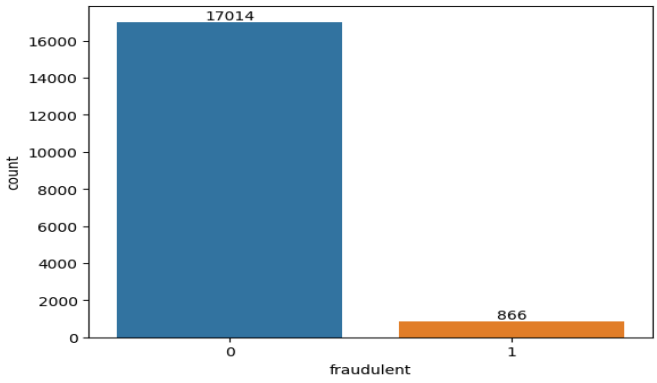

Figure 1: Class Ratio of Fraudulent Job Listings

There are significantly more non-fraudulent postings than fraudulent ones, making this dataset imbalanced. We will account for it by performing Stratified Sampling.

2.2.2 Salary

There is no distinctive distribution of the salary lower and upper bound. Upon closer inspection, the numbers are too varied, ranging from 0 to 8.000000e+08 for the lower bound and 0 to 1.200000e+09 for the upper bound.

2.2.3 Categorical Columns

Based on the analysis of various categorical variables with `fraudulent`, the charts consistently show a similar pattern - likely due to the imbalanced dataset. Hence, there is no obvious pattern that strongly correlates with fraud incidence, suggesting that it would be prudent to conduct a more detailed exploration to uncover more complex patterns.

2.2.4 Statistical Analysis of Textual Columns

In examining the distributions of word, character and sentence counts for `company_profile` and `requirements`, there is a noticeable difference between 'fraudulent' and 'non-fraudulent'. This contrast is also evident in `function`, where the distribution of word and character counts again varies. More EDA can be

2.3 Splitting of data - Handling of Imbalanced data

In addressing the highly imbalanced dataset, we utilised Stratified Sampling during the dataset split into training and test sets. This technique ensures that the original distribution of the 'fraudulent' target variable is maintained in both datasets, mitigating the risk of model bias toward the majority class. The use of sklearn's train_test_split with the `stratify=y` parameter achieved this stratified splitting, as we allocated 80% of the data to training and 20% to testing.

2.4 Handling of Text Data

Post dataset split, we enhanced text processing by consolidating key columns (["title", "company_profile", "department", "description", "requirements", "benefits", "industry", "function"]) into a unified 'combined_text' feature in both sets. This consolidation streamlines natural language processing (NLP) and feature extraction, providing a comprehensive representation of job postings for improved model training and evaluation.

3 Data Preprocessing and Model Selection

We will explore running different word embedding methods with different supervised machine-learning models.

To evaluate our models, we will use the F1 score and the Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) over accuracy as performance metrics due to the imbalanced nature of the data and the cost of different types of errors. With our imbalanced dataset, a model will return a high accuracy score by simply predicting the majority class (non-fraudulent) for all instances. As our goal is fraud detection, this model would be useless.

False positives (legitimate job listings flagged as fraudulent) and false negatives (fraudulent listings not being flagged) carry similar costs in our context. False positives might lead to legitimate listings being unfairly removed, causing frustration for real users and impacting the user experience. On the other hand, false negatives allow fraudulent listings to stay posted, deceiving users and harming the platform's credibility. Both result in a loss of

trust in the platform, potentially impacting user retention and the platform's reputation. Therefore, finding a balance between precision and recall becomes essential in this context, as the cost and implication of both errors are comparably large. Thus, the F1 score is useful here, as it provides a single metric that balances these two factors, ensuring that efforts to reduce one type of error don't disproportionately increase the other.

AUC is not affected by the imbalance in the classes, it provides a measure of how well the model can distinguish between the two classes, independent of the threshold set for classification.

3.1 Term Frequency - Inverse Document Frequency (TF-IDF)

3.1.1 Data Preprocessing

Scaling of Numerical Columns:
First, we processed the salary range columns in the dataset to a consistent range, typically between 0 and 1, to ensure that their magnitudes do not disproportionately influence machine learning models. This is done by using StandardScaler from sklearn, by standardising features by removing the mean and variance and scaling it to unit variance.

Ordinal and Frequency Encoding:
Next, we processed the categorical features either through Ordinal or Frequency Encoding. For the categorical feature `location_frequency`, due to its high cardinality, we performed frequency encoding, replacing the categorical values with their corresponding frequencies in the dataset, offering a representation that reflects the prevalence of each category. For the other categorical features, we performed ordinal encoding using sklearn's OrdinalEncoder on categorical variables such as required_experience and required_education, assigning numerical values to categories based on their inherent order, preserving the ordinal relationships between categories.

Text Processing:
Processing the textual column involves a series of steps to enhance the quality and relevance of textual data. Such steps include converting text to lowercase, removing square bracketed content, eliminating hyperlinks, discarding HTML tags, erasing punctuation, and filtering out words containing numbers. Furthermore, we removed English stopwords, reducing noise and focusing on essential content. Lastly, we created a lemmatizer function that utilises WordNet lemmatisation to standardise words to their base forms, aiding in maintaining consistency across the vocabulary. This comprehensive approach ensures a standardised and refined representation of the text, optimising it for subsequent natural language processing (NLP) tasks and machine learning analyses.

Vectorise processed text using TF-IDF:
Lastly, we performed text vectorization using TF-IDF, which converts processed text into numerical vectors. TF-IDF takes into account the frequency of words in a document relative to their occurrence across the entire corpus. This results in a numerical representation that captures the importance of words in individual documents, providing a suitable input for our machine learning models to analyse and learn from.

3.1.2 Random Forest

Random Forest is a supervised machine learning model that uses several decision trees to aid in prediction-making and is hence less likely to overfit than decision trees because it combines the predictions from multiple trees.
It also generally requires less hyperparameter tuning compared to algorithms like SVM.
Model Performance

We run the Random Forest on both the Training and Test sets to achieve the performance metrics below. Confusion matrices for both the training set and test set can be found in the appendix (Figure B1).

```
Metrics for Training Set:
Accuracy: 0.9516
AUC: 0.9744

Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98     13611
           1       0.00      0.00      0.00       693

    accuracy                           0.95     14304
   macro avg       0.48      0.50      0.49     14304
weighted avg       0.91      0.95      0.93     14304
```

```
Metrics for Test Set:
Accuracy: 0.9516
AUC: 0.9640

Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98      3403
           1       0.00      0.00      0.00       173

    accuracy                           0.95      3576
   macro avg       0.48      0.50      0.49      3576
weighted avg       0.91      0.95      0.93      3576
```

Figure 2. Performance Metrics for Training and Test Data for Random Forest

Insights

Random Forest's advantage is that it excelled in identifying non-fraudulent job postings, as reflected in excellent precision, recall, and f1-score for the non-fraudulent class. It also demonstrated a high AUC score for both the training and test datasets. However, as observed from confusion matrices of both the training and test data, it failed to predict fraudulent job postings, resulting in zero recall, precision and f1-score for the fraudulent class. This overly conservative model has a tendency to misclassify non-fraudulent postings as fraudulent as seen by the low recall.

In light of such a poor performance in detecting fraudulent cases, attempts were made to use Principal Component Analysis (PCA) for dimensionality reduction. However, the nature of the data being too sparse made the implementation of it challenging. The data is mainly populated with non-fraudulent cases to contain a large number of zero values. This sparsity complicates the extraction of meaningful principal components.

3.1.3 Logistic Regression

Logistic Regression is a supervised learning algorithm utilised for binary classification tasks. The model estimates the probability of an observation falling into the positive class by applying the logistic function to a linear combination of input features. In the classification process, a decision threshold is set, and instances with predicted probabilities above the threshold are assigned to the positive class, while those below the threshold are assigned to the negative class.

Model Performance

Confusion matrices for both the training set and test set can be found in the appendix (Figure B2).

```
Metrics for Training Set:
Accuracy: 0.9765
AUC: 0.9934

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     13611
           1       0.99      0.52      0.68       693

    accuracy                           0.98     14304
   macro avg       0.99      0.76      0.83     14304
weighted avg       0.98      0.98      0.97     14304
```

```
Metrics for Test Set:
Accuracy: 0.9757
AUC: 0.9853

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3403
           1       0.99      0.50      0.67       173

    accuracy                           0.98      3576
   macro avg       0.98      0.75      0.83      3576
weighted avg       0.98      0.98      0.97      3576
```

Figure 3. Performance Metrics for Training and Test Data for Logistic Regression

Insights

Both fraud and non-fraud data do not show an increase in the f1-score from the training set to the test set. However, the test set shows a small rise in fraudulent data recall. The precision in predicting fraudulent cases decreases from training data to test data. Moreover, since the test and training AUC and F1-score are similar, our model is not overfitted. While the model showcases a high AUC (0.9757), it struggles with the precision-recall balance for the fraudulent class, demonstrated by its f1-score (0.67)). Its ability to identify non-fraudulent postings, highlighted by its precision (0.98), recall (1.00) and f1 (0.99) metrics, contrasts sharply with its difficulties in correctly classifying fraudulent postings, reflecting the model's struggles with imbalanced data. Hence, a linear approach may not adequately handle the complexity inherent in fraudulent cases.

3.1.4 Support Vector Machine

Support Vector Machines (SVM) is a supervised machine learning model employed for binary classification tasks. It finds a hyperplane in the feature space that maximally separates the two classes and is determined by support vectors. The prediction for a new instance is based on which side of the hyperplane it falls.

Model Performance

Confusion matrices for both the training set and test set can be found in the appendix (Figure B3).

Metrics for Training Set:
Accuracy: 0.9819
AUC: 0.9995

Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	13611
1	1.00	0.63	0.77	693
accuracy			0.98	14304
macro avg	0.99	0.81	0.88	14304
weighted avg	0.98	0.98	0.98	14304

Metrics for Test Set:
Accuracy: 0.9796
AUC: 0.9935

Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	3403
1	1.00	0.58	0.73	173
accuracy			0.98	3576
macro avg	0.99	0.79	0.86	3576
weighted avg	0.98	0.98	0.98	3576

Figure 4. Performance Metrics for Training and Test Data for SVM

Fine-tuning Hyperparameters for SVM

Observing the low f1-score the SVM model achieves on the test data (0.73 for fraudulent), Cross Validation is carried out on the training data to determine the optimal hyperparameters used for the SVC model. We used GridSearchCV() with Stratified k-Fold as our data is imbalanced, which ensures that each fold has a similar distribution of the Fraudulent column. After splitting the training data into k subsets, all k-1 subsets are iteratively treated as training data and evaluated on the remaining subsets. This gives us recommendations for the optimal hyperparameters to use in our model.

However, it is pertinent to note that this process is computationally expensive. Due to the large time complexity involved, the process took more than two hours to complete. Parameter values tested were as follows: {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf'], 'gamma': ['scale', 'auto']}. Optimal values recommended, and subsequently run on the test data, are as follows: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}.

```
Metrics for Test Set:
Accuracy: 0.9888
AUC: 0.9935

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3403
           1       0.98      0.79      0.87       173

    accuracy                           0.99      3576
   macro avg       0.98      0.89      0.93      3576
weighted avg       0.99      0.99      0.99      3576
```

Figure 5. Performance Metrics for Test Data after fine-tuning hyperparamters for SVM

There is an improvement in the f1-score of fraudulent cases from 0.73 to 0.87 with cross-validation. Confusion matrices for the test data after cross-validation can be found in the appendix (Figure B4).

Insights

After running with optimal hyperparameters, the SVM demonstrated excellent performance. The high AUC score (0.9935) highlights the model's ability to discriminate between classes. Furthermore, the SVM model achieves near-perfect precision, recall, and f1-score for non-fraudulent job postings. With a reduced difference in AUC from the Training set to the Test set as a consequence of cross-validation, our model is more generalised and less prone to overfitting. Yet, there continue to be challenges in correctly identifying fraudulent postings, with a lower recall and F1-score for the fraudulent class in the new Test set compared to the Training set.

3.1.5 XGBoost

XGBoost is a supervised learning algorithm and an ensemble learning technique that consecutively builds a series of decision trees. It integrates the predictive ability of several poor learners. Because it provides effective training, regularisation strategies, and adaptability to handle intricate relationships within the data, XGBoost generally outperforms most other models.

Model Performance

Confusion matrices for both the training set and test set can be found in the appendix (Figure B5).

```
Metrics for Training Set:
Accuracy: 0.9999
AUC: 1.0000

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13611
           1       1.00      1.00      1.00       693

    accuracy                           1.00     14304
   macro avg       1.00      1.00      1.00     14304
weighted avg       1.00      1.00      1.00     14304
```

```
Metrics for Test Set:
Accuracy: 0.9866
AUC: 0.9940

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3403
           1       0.97      0.75      0.84       173

    accuracy                           0.99      3576
   macro avg       0.98      0.87      0.92      3576
weighted avg       0.99      0.99      0.99      3576
```

Figure 6. Performance Metrics for Training and Test Data for XGBoost

Insights

The model has performed well across both training and testing, showcasing a high AUC (0.9940) in testing. It also maintains near-perfect precision (0.99) and F1-score (0.99) for non-fraudulent cases in testing, indicating its robustness. Although there is a drop in the recall (0.75) for fraudulent cases, the model still presents strong predictive power with high precision (0.97), minimising false positives. While the model could be overfitted as seen by the F1-score and AUC of 1.00, the model still performs relatively well on the test set, achieving f1 score of 0.84 on the fraudulent class and and AUC of 0.994.

3.2 Enhanced Representation through Knowledge Integration (ERNIE)

Next, we explored using ERNIE to encode processed text into context-aware embeddings. Using its pre-trained contextual understanding, ERNIE transformed each processed text into high-dimensional numerical representations, capturing nuanced semantic relationships.

3.2.1 ERNIE for Text Classification

Model Performance

Metrics for Training Set:
Accuracy: 0.9865
AUC: 0.9588

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	13611
1	0.95	0.76	0.85	693
accuracy			0.99	14304
macro avg	0.97	0.88	0.92	14304
weighted avg	0.99	0.99	0.99	14304

Metrics for Test Set:
Accuracy: 0.9821
AUC: 0.9460

Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	3403
1	0.91	0.70	0.79	173
accuracy			0.98	3576
macro avg	0.95	0.85	0.89	3576
weighted avg	0.98	0.98	0.98	3576

The ERNIE model exhibits strong performance with a training f1 score of 0.85 and an AUC of 95.88%. It maintains a high f1 score on the test set at 0.79 with an AUC of 94.60%, indicating minimal drop-off and good generalisation from training to unseen data. Precision for non-fraudulent postings is consistent, with a marginal decrease in recall for fraudulent cases from training to testing.

Insights

The ERNIE model demonstrates excellent capability in distinguishing non-fraudulent postings, mirrored by high precision and recall. The model's performance on fraudulent postings, characterised by high precision, indicates a strong ability to correctly identify fraud when it does. A slight decrease in recall for the fraudulent class from training to test does not significantly impact the overall performance, suggesting the model is robust. The minuscule difference in recall and F1 score indicates that the model is not overfitted.

Conclusion

The minimal performance drop from training to test sets is indicative of the model's effective learning and generalisation capabilities, making it a reliable choice for detecting fraudulent job postings. Further refinement could enhance recall for the fraudulent class without compromising precision.

4 Evaluation of Models

4.1 Model Comparison

4.1.1 TF-IDF Vectorisation

These are the metrics for the “fraudulent” class for the models we ran after using TF-IDF vectorisation.

Model		**Precision**	**Recall**	**F1-Score**	**AUC**
Random Forest	Train Set	0.00	0.00	0.00	0.9744
	Test Set	0.00	0.00	0.00	0.9640
Logistic Regression	Train Set	0.99	0.52	0.68	0.9934
	Test Set	0.99	0.50	0.67	0.9853
Support Vector Machine	Train Set	1.00	0.63	0.77	0.9995
	Test Set	**0.98**	**0.79**	**0.87**	**0.9935**
XGBOOST	Train Set	1.00	1.00	1.00	1.0000
	Test Set	0.97	0.75	0.84	0.9940

Between the models trained on the TF-IDF-vectorised Training data, Support Vector Machine after cross-validation achieves the highest F1-score on the Test set of 0.87. This indicates that it has the best balance between precision and recall.

However, it is pertinent to note that this is a Fraud Classification model. Minimising false positives, or real transactions that are incorrectly classified as fraudulent, could be critical. Thus, a model with higher precision might still be preferable, even if it comes at the cost of lower recall. The Logistic Regression model is most suited for this case, boasting a high precision on the test set of 0.99 despite having a lower recall.

4.1.2 ERNIE Word Embeddings

These are the metrics for the “fraudulent” class for the model we ran after using ERNIE for vectorisation with a classification layer on top of it. Confusion matrices for both the training set and test set can be found in the appendix (Figure B6).

Model		**Precision**	**Recall**	**F1-Score**	**AUC**
ERNIE	Train Set	0.95	0.76	0.85	0.9588
	Test Set	0.91	0.70	0.79	0.9470

4.2 Overall Comparison

Comparing between the combined use of TF-IDF for vectorisation with SVM with cross-validation as a model and ERNIE for vectorisation with a classification layer on top of it to fine-tune the model for fraud detection, it is clear that the former achieves higher precision (0.98), recall (0.79), and f1-score (0.87). Having a high precision is crucial in fraud detection where minimising false positives is a priority. It is thus the most preferable model we can use in detecting fraudulent job postings.

4.3 Model Limitation

All models show a slight drop in AUC from the train to test set, indicating a consistent pattern across different algorithms. This suggests that the issue is not related to any specific model's overfitting, but rather an inherent characteristic of the dataset and the complexity of the fraud detention domain. It is important to note that fraudulent behaviour is not static. Instead, it evolves over time with subtle changes. A drop in performance is to be expected in unseen test data, especially in the context of fraud detection where new patterns can emerge that the model has not learned.

4.4 Comparison to Benchmark

The highest AUC score achieved for the same dataset by using the AutoGluon framework was 0.998 (Grover et al., 2022). Even though our best AUC score was achieved by using the SVM model which only gave an AUC score of 0.9915, our model still managed to outperform several other AutoML frameworks. We believe that the gap in performance between our models and the benchmark of 0.998 could be improved via several methods such as engineering of new features, re-sampling techniques and cost-sensitive learning.

Our best model is the combined use of TF-IDF for vectorisation with SVM with cross-validation as a model, having an f1 score of 0.86. According to existing literature, a general rule of thumb for a good F1 score is 0.7. Our models generally outperformed this benchmark, exceeding industry standards, except for the Logistic Regression models.

5 Conclusion

5.1 Application in Business Context

We believe that job posting sites such as Indeed, LinkedIn and Glassdoor can benefit from our insights and integrate our machine learning model into their business. As the models produced a competitive AUC score and a high f1-score for fraudulent cases, this machine model can be used as an initial step in identifying very high potential fraudulent job postings. This would improve the trust and safety of their platform, while also reducing the amount of manpower required operationally to check through the job postings manually. Furthermore, since the model may not be able to spot 100% of fraudulent job postings, these platforms can indicate to job seekers potential risky postings, to help them in making an informed decision. Overall, this model can improve the efficiency of these job listing sites in detecting fraudulent job listings and improve the trust and safety of their users.

5.2 Extension for Future Work

As mentioned in Chapter 3.1.2, Although Random Forest showed significant advantages in correctly classifying non-fraudulent job posts, it had limits when it came to handling the imbalanced data, producing poor metrics for the

fraudulent cases. Principal Component Analysis (PCA) was attempted to reduce the dimensionality of the dataset. However, the difficulties caused by the data's sparsity hindered PCA abilities. Thus, as an extension, additional study and experimentation using different optimisation techniques created especially to address the complexities of imbalanced datasets can be carried out. This may improve the system's capacity to detect fraud while using Random Forest. Overall, fine-tuning and optimization strategies may further enhance our models' abilities to identify fraudulent job postings, ensuring a more balanced and reliable classification across both classes.

6. Appendix

Appendix A. Features observed in the Dataset and their description

Feature	Explanation
job_id (int)	Unique identifier for each job posting
title (string)	The title or name of the job position
location (string)	Geographical location where the job is based
department (string)	The department within the company that is posting the job
salary_range (string)	The range of salary offered for the position
company_profile (string)	Description of the company
description (string)	Detailed information about the job responsibilities
requirements (string)	Qualifications needed for the job
benefits (string)	Additional perks provided with the job
telecommuting (int)	Whether the job allows working remotely, represented as 0 or 1
has_company_logo (int)	Whether the company's logo is displayed on the job posting, represented as 0 or 1
has_questions (int)	Whether there are screening questions as part of the application process, represented as 0 or 1
employment_type (string)	Type of employment, e.g., full-time or part-time
required_experience (string)	Level of experience needed for the job
required_education (string)	Education qualifications required for the job
industry (string)	Industry sector the job belongs to
function (string)	Functional area of the job, e.g., marketing, engineering
fraudulent (int)	Whether the job posting is identified as fraudulent, represented as 0 or 1

Appendix B. Confusion Matrices for Training and Test Data

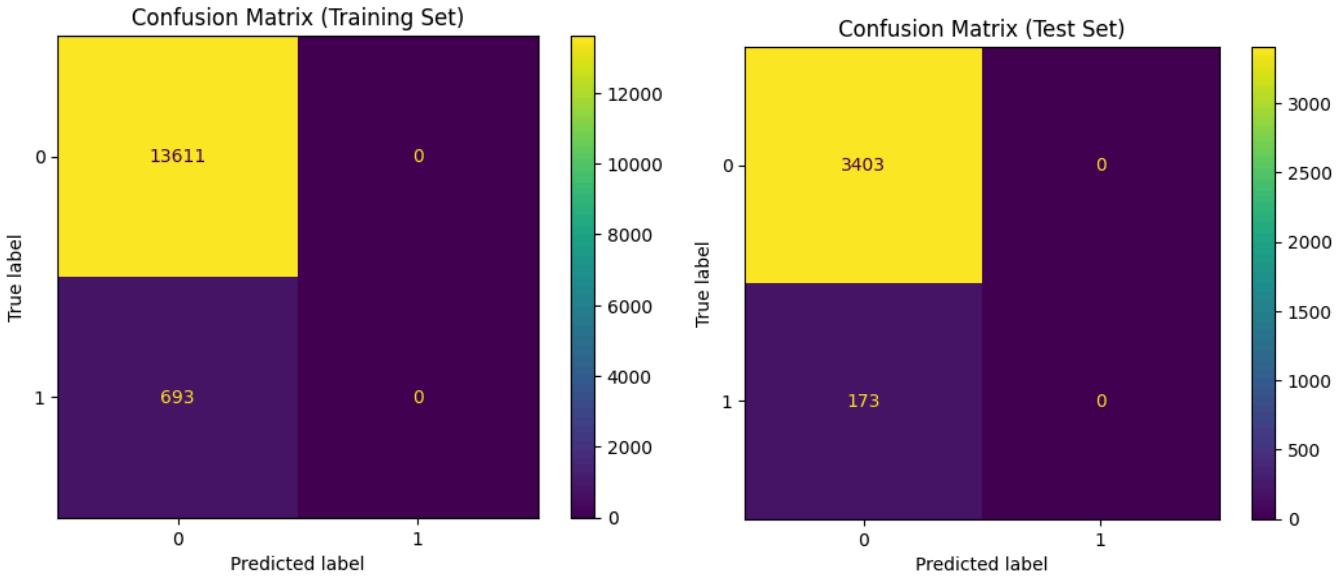

Figure B1. Confusion Matrices for Random Forest on Training and Test Data

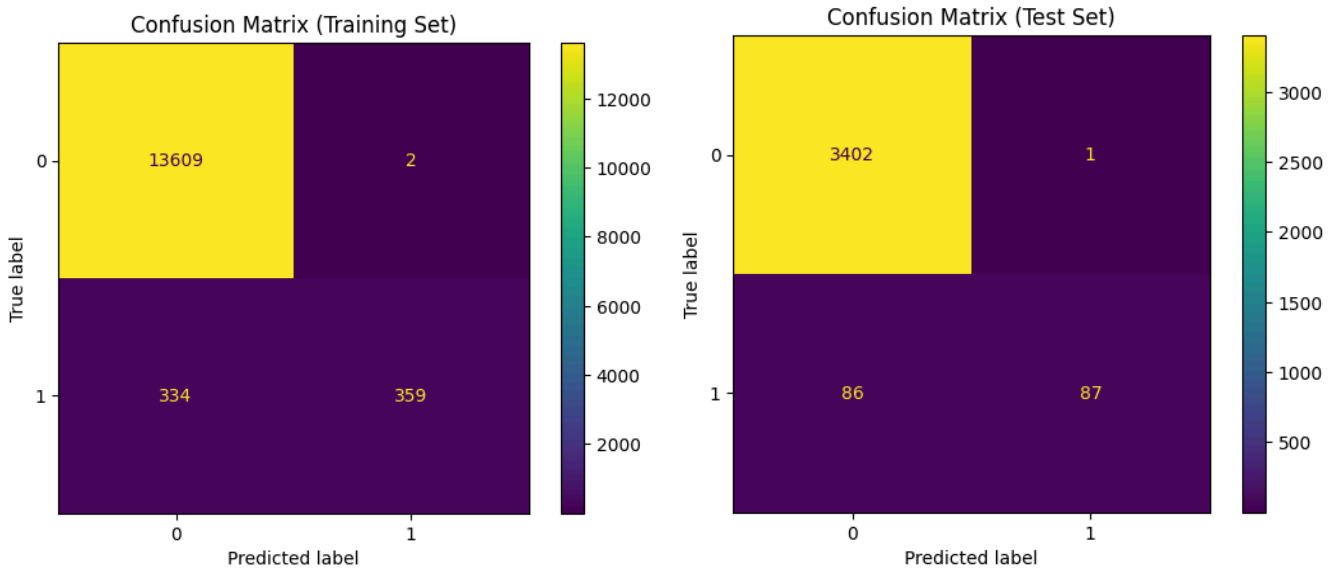

Figure B2. Confusion Matrices for Logistic Regression on Training and Test Data

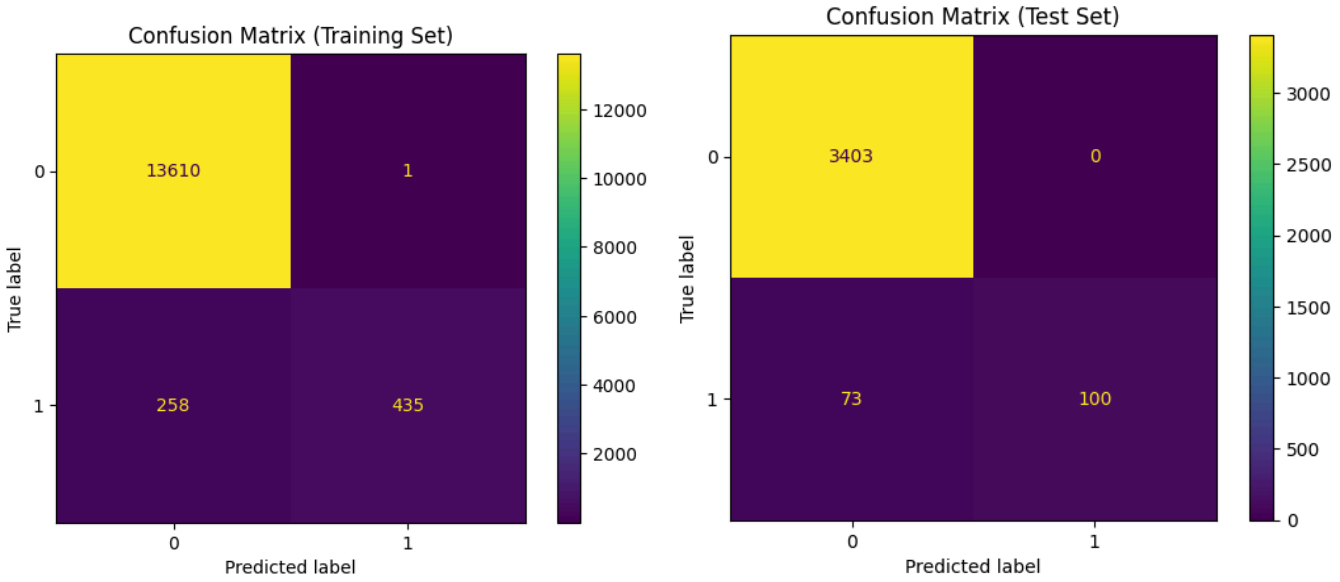

Figure B3. Confusion Matrices for SVM on Training and Test Data

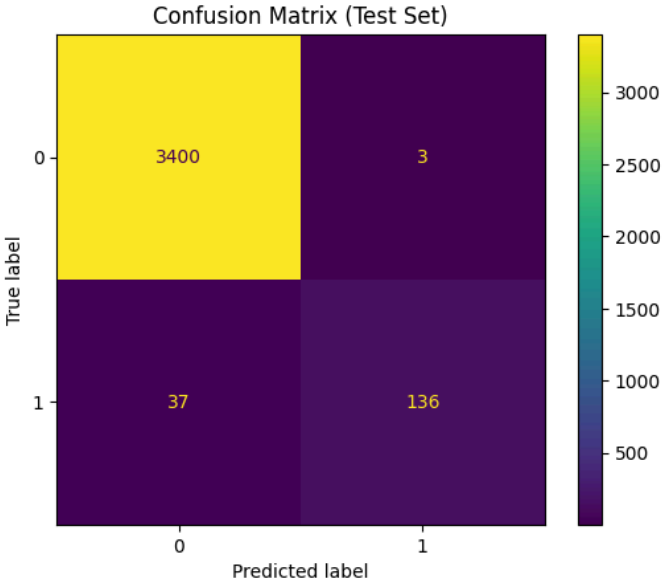

Figure B4. Confusion Matrices for Cross-Validated SVM on Test Data

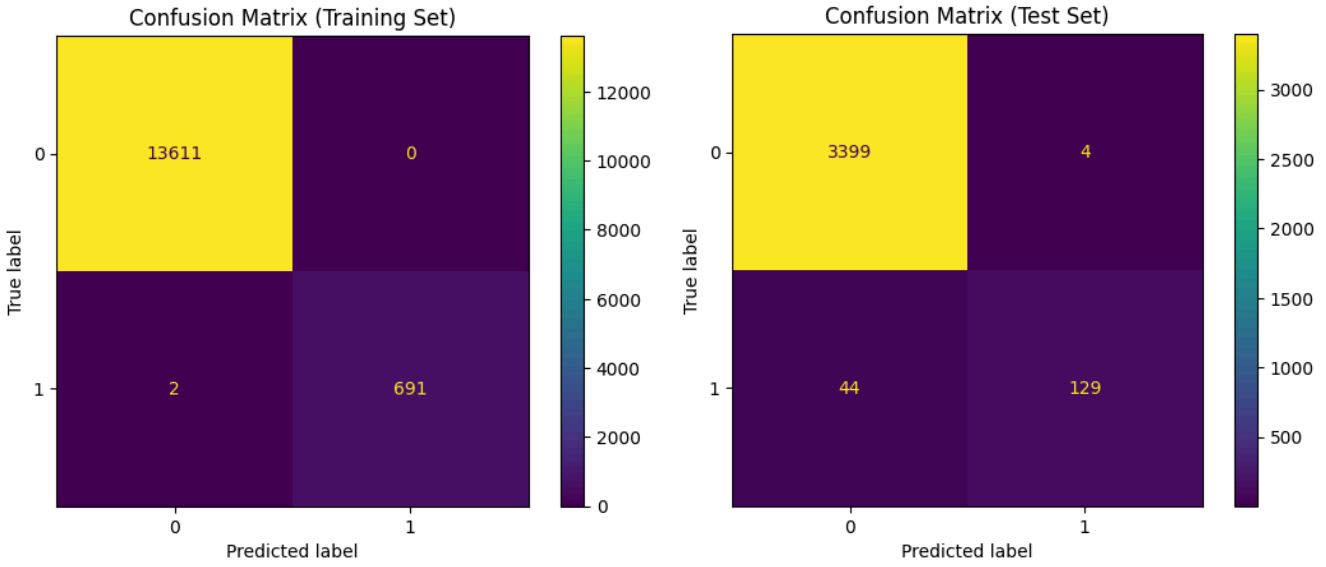

Figure B5. Confusion Matrices for XGBoost on Training and Test Data

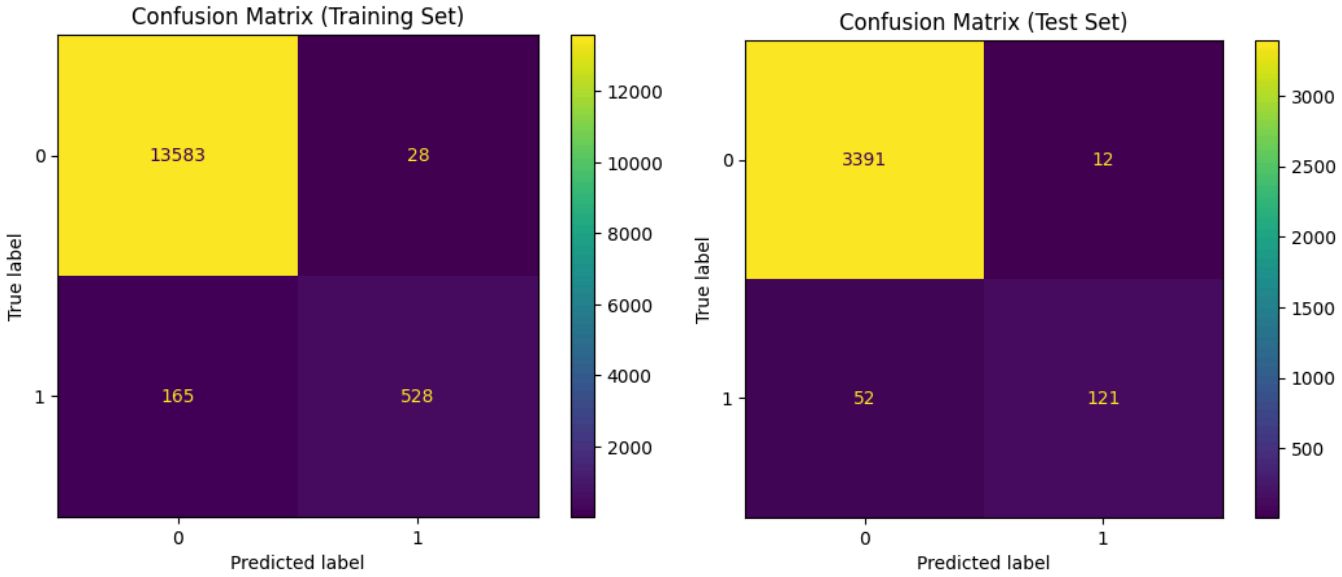

Figure B6. Confusion Matrices for ERNIE Classifier on Training and Test Data

References

1. Grover, P., Xu, J., Tittelfitz, J., Cheng, A., Li, Z., Zablocki, J., Liu, J., & Zhou, H. (2022). *Fraud Dataset Benchmark and Applications*, 5–5. https://doi.org/doi.org/10.48550/arXiv.2208.14417