BT4012 Fraud Analytics
Final Report
Group 26

Student Name	Student Matric Number
Lim Zhen Yong	A0236495U
Lo Zhi Hao	A0236437B
Lam Wen Jett	A0234935Y
Ng Han Leong, Jordan	A0233839W

1 Problem

In the post-COVID digital era, online job portals have become essential for connecting job seekers and employers, but they also attract fraud, such as fake job postings, leading to significant financial losses and a tainted job-seeking experience. In the first quarter of 2022 alone, the US recorded over 20,700 cases of job-related fraud, with a third resulting in monetary losses [1]. This problem, if unchecked, threatens to disrupt the job market and affect multiple parties involved.

Fraudulent job postings not only harm individuals seeking employment but also burden employers with inefficient recruitment processes. Reducing these fake listings can streamline hiring, boosting trust in online job searches. Therefore, platforms that effectively filter out scams will likely outperform competitors, retain more users, and potentially increase revenue.

Our project focuses on developing a fraud detection system using the Employment Scam Aegean Dataset (EMSCAD) to identify fraudulent job postings. This system is crucial for enhancing the reliability of online job markets, benefiting both job seekers and employers, and serves as a foundational step in addressing this widespread issue.

2 Data Handling

2.1 Data Source

We utilized the Employment Scam Aegean Dataset (EMSCAD), a publicly available dataset with 17,880 real-life job ads, extracted from Kaggle. The dataset consists of authentic job ads from Workable, providing a strong foundation for developing and testing resilient job fraud detection systems. EMSCAD covers job ads from 2012 to 2014, with each entry manually annotated by Workable employees through quality assurance procedures. Classification criteria were established based on factors like suspicious activity, false information, candidate complaints, and in-depth client analysis.

2.2 Data Description

This dataset, consisting of 17,880 job listings worldwide, encompasses positions across the United States of America (USA), Europe, and Asia. The primary objective of the dataset is to facilitate a classification task focused on distinguishing between authentic and deceptive job postings. Upon closer examination, we observe a significant class imbalance, with 17,014 instances of non-fraudulent job postings and only 866 instances of fraudulent ones, as illustrated in Figure 1. This results in a substantial 95% representation of the non-fraud class and only 5% representation of the fraud class. The disproportionate distribution may hinder the model's ability to effectively learn patterns and trends within the fraud class, requiring careful consideration during model development to ensure robust and unbiased results.

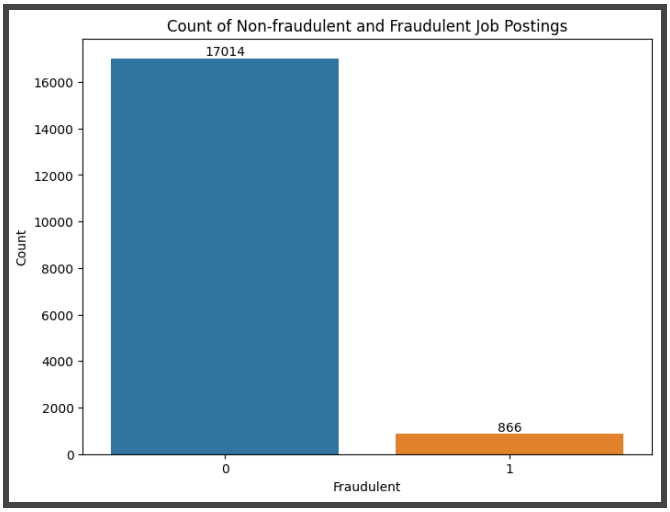

Figure 1. Count of non-fraudulent and fraudulent job postings

The dataset comprises 18 features, with the 'fraudulent' feature serving as the binary target variable for classification purposes. Of the remaining features, three of them are also binary in nature, which are 'Telecommuting', 'Has Company Logo', and 'Has Question', while the remaining features are either categorical strings or text strings, suggesting a need for preprocessing or encoding techniques. These features are indicated in Figure 2, which also includes the number of NAs values found within that feature.

Feature	**Type of feature**	**Description**	**Number of NAs**
job_id	int64	Unique ID value	0
title	string	Categorical value of the title of the job	0
location	string	Categorical value of the location of job	346
department	string	Categorical value of the department of job	11547
salary_range	string	Range of salary, from the lower to upper range	15012
company_profile	string	Description of the company hiring	3308
description	string	Description of the job offered	1
requirements	string	Description of the requirements of the job, includes educational qualifications, duration, soft and hard skills	2695
benefits	string	Description of the job benefits	7210
telecommuting	int64	Binary value to indicate whether the job allows for telecommuting	0
has_company_logo	int64	Binary value to indicate whether the company has a logo picture	0

has_questions	int64	Binary value to indicate whether the job listing has screening questions	0
employment_type	string	Categorical value indicating the type of employment, such as Full-Time, Part-Time etc	3471
required_experience	string	Categorical value indicating the type of past work experience required, such as Entry Level, Internship, Mid-Senior level etc	7050
required_education	string	Categorical value indicating the type of educational qualification required, such as Bachelor's Degree, High School etc	8105
industry	string	Categorical value indicating the industry the company is in, such as Financial Services, Information Technology and Services, Events Services etc	4903
function	string	Categorical value indicating the function within the company, such as Customer Service, Information Technology, Engineering etc	6455
fraudulent	int64	Binary value indicating whether the post is fraudulent or not, where 1 represents fraudulent postings (positive class) and 0 indicating non-fraudulent postings (negative class)	0

Figure 2. Table of dataset features

The dataset exhibits a notable prevalence of missing values across various fields, thereby necessitating proper data engineering strategies to address these gaps. Missing data can significantly impact the overall integrity and representativeness of the dataset, potentially leading to biased analyses and unreliable model predictions. Furthermore, many algorithms require complete datasets to operate effectively, and the presence of missing values may impede their performance or lead to suboptimal results. In addition, imputing or removing missing values is essential for optimizing the predictive capabilities of models, as it enables them to learn patterns and relationships from a more comprehensive and complete set of data, ultimately contributing to the robustness and reliability of any subsequent analyses or applications.

Hence, it is crucial to navigate the dataset's inherent challenges, including the imbalanced class distribution and the widespread occurrence of missing data, with careful consideration. This is essential to conduct a comprehensive evaluation of the dataset and ensure a thorough understanding of its potential impacts on analyses or model development.

2.3 Exploratory Data Analysis

From our Exploratory Data Analysis, we discovered several key insights. The top insights that we have compiled are as follows:

- Positions that require `Executive` and `entry-level` experiences have the highest chance (percentage) of being a fraudulent posting.

- Jobs with `Administrative`, `Financial Analyst`, and `Accounting/Auditing` job functions have the highest chance (percentage) of being fraudulent postings.
- `Part-time` positions have a higher chance (percentage) of being fraudulent job postings as compared to other employment types.
- Fraudulent postings tend to use the word `project` more frequently as compared to non-fraudulent ones.
- Fraudulent postings tend to have shorter or no company profiles. There is a higher chance (percentage) of fraudulent cases when `company_profile` is not available. Meanwhile, the mean of `company_profile` for fraudulent posts is 33 words long as compared to 95 words for non-fraudulent ones.

2.4 Data Preprocessing

We utilized various data preprocessing techniques in order to ensure data quality, engineer useful features, and also ensure compatibility between the features and algorithms that we are using. Below are the tasks that we performed using data preprocessing techniques:

Handling Missing Values

As stated earlier, our dataset contains a sizable number of missing values in various columns. Thus, the handling of missing values is crucial in ensuring high performance of models.

In particular, after performing Exploratory Data Analysis in all the columns, we decided that the most logical and efficient way to handle the missing values is to classify the missing values as `unknown` for categorical values and `no available data` for text data. As we experienced a lack of context regarding how the data were collected and the details of the job postings, we decided that it would be best to treat the missing values as MCAR (Missing Completely At Random) and thus deal with the missing values via our method stated earlier.

The Handling Missing Values data preprocessing step is applied to all columns with NA values observed in the dataset.

Column-Specific Feature Engineering

We performed column-specific feature engineering on certain columns to improve model performances and also ensure that the model would not misinterpret and overfit the data points. In particular, the approach we took to handle the `location` column and `department` column is worth mentioning.

The `location` column consists of the country code and the city the job posting is in. To further extract the information from the column, we decided to split the `location` column into `country` and `city` columns. Moving forward, we performed one hot encoding on both columns to encode it into something the models could interpret.

Meanwhile, the `department` column consists of 1,337 unique departments, where some departments are very rare with only 1 or 2 observations. To ensure that our models do not

overfit this issue and reduce the dimensionality of our training data, we decided to categorize departments that are very rare as `others`. This is distinct from the `unknown` class we used when we filled in the NA values in the column, as we believe that the rare departments may capture certain hidden information in the data. Similar operations are applied to the `country`, `city`, and `industry` columns.

Removing Stopwords, Tokenization, Stemming and Lemmatization for text data

Based on our Exploratory Data Analysis, we noticed that the dataset contains a significant amount of text data. Furthermore, after conducting further visualizations and analysis, we believe that capturing the context in the text data provided would help improve our model's performance significantly by providing more high-value features. Thus, data preprocessing has been conducted on the dataset to ensure that we are able to capture the context in the text data effectively.

The text data in the dataset is mainly from columns `title`, `company_profile`, `description`, 'requirements` and `benefits`. For ease of analysis, we decided to concatenate all text data columns into one major column, `full_text`.

In terms of text preprocessing methods, first and foremost, stopwords that lack significant meaning are removed. This step is essential to reduce noise in our text data and improving model efficiency. Next, tokenization is conducted to break down text data into individual words. Stemming and lemmatization are further implemented to normalize the words into their most basic forms for future operations.

Bag of Words (BoW), Word Embeddings and other potential NLP methods to transform text data into machine-readable format

A few methodologies have been experimented with to explore the best way of preprocessing the data. In particular, Bag of Words (BoW) methodologies such as CountVectorizer and TfIdfVectorizer are utilized to come up with statistical embeddings of the vocabularies in the text.

On the other hand, pretrained Word2Vec Word Embedding was also experimented with in order to identify methods of preprocessing that led to the best features.

SMOTE Oversampling

To handle the imbalanced nature of the dataset, we decided to explore the use of Synthetic Minority Oversampling Technique (SMOTE) in our train set in order to improve model performances, prevent model skewing, and enhance the generalisability of the models.

2.5 Implementation of Classifiers and Models

For the problem statement on hand, we decided to frame it into a binary supervised classification problem, where we are trying to predict the `fraudulent` class in the data.

Attached is the list of models we decided to use for our model training:

1. Logistic Regression
2. Decision Tree Classifier
3. Support Vector Machines
4. K Nearest Neighbors
5. ExtraTree Classifier
6. Random Forest
7. XGBoost
8. ADABoost
9. ExtraTrees Classifier
10. Bidirectional LSTM

We experimented with 10 machine learning models in total, and our aim is to ensure that we are able to find the best-performing models that best suit the problem statement. The best classifier would be chosen to have outstanding performance among all the peer classifiers for each feature set.

A train-test split is performed for all model training and evaluation. For all 10 models, we utilized the same train and test sets by setting the same random seed consistently, thus ensuring the same orientation of splitting the data points. The same data preprocessing techniques are used for all models as well, to ensure the data provided to each model remains consistent.

3 Model Performance Evaluation

In terms of evaluation metrics, a confusion matrix was used, together with a recording of Accuracy, Precision, Recall, F1-Score, and Roc AUC score for the model. These metrics are selected in order to provide a more robust and complete overview of the model's performance.

The performance of the 10 models before and after oversampling is recorded as below:
(highlighted are the best optimal scores recorded)

Model Name	**Accuracy**	**F1-Score**	**Precision**	**Recall**	**Roc AUC**
Logistic Regression	CountVec: 98% Word2Vec: 97% TF-IDF: 97%	CountVec: 82% Word2Vec: 70% TF-IDF: 63%	CountVec: 90% Word2Vec: 84% TF-IDF: 87%	CountVec: 75% Word2Vec: 60% TF-IDF: 49%	CountVec: 97% Word2Vec: 95% TF-IDF: 97%
KNN Classifiers	CountVec: 94% Word2Vec: 98% TF-IDF: 98%	CountVec: 56% Word2Vec: 73% TF-IDF: 75%	CountVec: 45% Word2Vec: 91% TF-IDF: 77%	CountVec: 73% Word2Vec: 61% TF-IDF: 74%	CountVec: 93% Word2Vec: 89% TF-IDF: 95%
Decision Tree Classifer	CountVec: 98% Word2Vec: 97% TF-IDF: 97%	CountVec: 78% Word2Vec: 71% TF-IDF: 72%	CountVec: 76% Word2Vec: 68% TF-IDF: 74%	**CountVec: 81%** Word2Vec: 73% TF-IDF: 71%	CountVec: 90% Word2Vec: 86% TF-IDF: 87%
ExtraTree Classifer	CountVec: 96% Word2Vec: 97% TF-IDF: 97%	CountVec: 62% Word2Vec: 64% TF-IDF: 69%	CountVec: 62% Word2Vec: 64% TF-IDF: 71%	CountVec: 63% Word2Vec: 64% TF-IDF: 68%	CountVec: 80% Word2Vec: 81% TF-IDF: 85%

RandomForest Classifer	CountVec: 98% Word2Vec: 97% TF-IDF: 98%	CountVec: 76% Word2Vec: 60% TF-IDF: 76%	CountVec: 100% Word2Vec: 100% TF-IDF: 92%	CountVec: 62% Word2Vec: 43% TF-IDF: 65%	CountVec: 99% Word2Vec: 99% TF-IDF: 98%
ExtraTrees Classifer	CountVec: 98% Word2Vec: 98% TF-IDF: 98%	CountVec: 81% Word2Vec: 75% TF-IDF: 77%	CountVec: 100% Word2Vec: 100% TF-IDF: 92%	CountVec: 67% Word2Vec: 60% TF-IDF: 66%	CountVec: 100% Word2Vec: 99% TF-IDF: 98%
SVC	CountVec: 98% Word2Vec: 98% TF-IDF: 97%	CountVec: 71% Word2Vec: 66% TF-IDF: 66%	CountVec: 99% Word2Vec: 99% TF-IDF: 97%	CountVec: 56% Word2Vec: 49% TF-IDF: 50%	CountVec: 98% Word2Vec: 95% TF-IDF: 98%
Adaboost	CountVec: 97% Word2Vec: 97% TF-IDF: 96%	CountVec: 66% Word2Vec: 53% TF-IDF: 50%	CountVec: 81% Word2Vec: 76% TF-IDF: 83%	CountVec: 55% Word2Vec: 41% TF-IDF: 35%	CountVec: 97% Word2Vec: 96% TF-IDF: 95%
XGB Classifer	CountVec: 98% Word2Vec: 98% TF-IDF: 98%	CountVec: 83% Word2Vec: 80% TF-IDF: 79%	CountVec: 94% Word2Vec: 98% TF-IDF: 92%	CountVec: 75% Word2Vec: 67% TF-IDF: 69%	CountVec: 99% Word2Vec: 99% TF-IDF: 98%
LSTM	CountVec: 99% Word2Vec: 99% TF-IDF: 98%	**CountVec: 87%** Word2Vec: 86% TF-IDF: 78%	CountVec: 97% Word2Vec: 91% TF-IDF: 76%	CountVec: 78% **Word2Vec: 81%** **TF-IDF: 81%**	CountVec: 81% Word2Vec: 90% TF-IDF: 90%

Figure 3: Performance on test set (Before Oversampling)

Model Name	**Accuracy**	**F1-Score**	**Precision**	**Recall**	**Roc AUC**
Logistic Regression	CountVec: 98% Word2Vec: 95% TF-IDF: 95%	CountVec: 81% Word2Vec: 56% TF-IDF: 62%	CountVec: 85% Word2Vec: 42% TF-IDF: 48%	CountVec: 78% Word2Vec: 84% TF-IDF: 86%	CountVec: 96% Word2Vec: 94% TF-IDF: 96%
KNN Classifiers	CountVec: 75% Word2Vec: 95% TF-IDF: 86%	CountVec: 28% Word2Vec: 61% TF-IDF: 42%	CountVec: 16% Word2Vec: 49% TF-IDF: 27%	CountVec: 96% Word2Vec: 80% **TF-IDF: 97%**	CountVec: 92% Word2Vec: 90% TF-IDF: 95%
Decision Tree Classifer	CountVec: 97% Word2Vec: 96% TF-IDF: 96%	CountVec: 72% Word2Vec: 66% TF-IDF: 65%	CountVec: 66% Word2Vec: 59% TF-IDF: 56%	CountVec: 79% Word2Vec: 76% TF-IDF: 78%	CountVec: 88% Word2Vec: 87% TF-IDF: 88%
ExtraTree Classifer	CountVec: 96% Word2Vec: 94% TF-IDF: 95%	CountVec: 65% Word2Vec: 54% TF-IDF: 60%	CountVec: 59% Word2Vec: 44% TF-IDF: 49%	CountVec: 72% Word2Vec: 68% TF-IDF: 77%	CountVec: 85% Word2Vec: 82% TF-IDF: 87%
RandomForest Classifer	CountVec: 98% Word2Vec: 98% TF-IDF: 97%	CountVec: 78% Word2Vec: 81% TF-IDF: 74%	CountVec: 89% **Word2Vec: 97%** TF-IDF: 67%	CountVec: 69% Word2Vec: 69% TF-IDF: 85%	CountVec: 99% Word2Vec: 99% TF-IDF: 98%
ExtraTrees Classifer	CountVec: 98% Word2Vec: 99% TF-IDF: 97%	CountVec: 81% Word2Vec: 84% TF-IDF: 76%	CountVec: 94% **Word2Vec: 97%** TF-IDF: 72%	CountVec: 71% Word2Vec: 75% TF-IDF: 80%	CountVec: 99% Word2Vec: 99% TF-IDF: 98%
SVC	CountVec: 98% Word2Vec: 97% TF-IDF: 97%	CountVec: 75% Word2Vec: 69% TF-IDF: 74%	CountVec: 86% Word2Vec: 61% TF-IDF: 71%	CountVec: 67% Word2Vec: 79% TF-IDF: 76%	CountVec: 97% Word2Vec: 97% TF-IDF: 97%
Adaboost	CountVec: 94%	CountVec: 57%	CountVec: 44%	CountVec: 83%	CountVec: 96%

	Word2Vec: 92% TF-IDF: 89%	Word2Vec: 50% TF-IDF: 46%	Word2Vec: 37% TF-IDF: 31%	Word2Vec: 79% TF-IDF: 90%	Word2Vec: 95% TF-IDF: 95%
XGBClassifier	CountVec: 98% Word2Vec: 99% TF-IDF: 96%	CountVec: 84% **Word2Vec: 87%** TF-IDF: 70%	CountVec: 87% Word2Vec: 93% TF-IDF: 59%	CountVec: 81% Word2Vec: 82% TF-IDF: 86%	CountVec: 99% Word2Vec: 98% TF-IDF: 98%
LSTM	CountVec: 98% Word2Vec: 98% TF-IDF: 98%	CountVec: 83% Word2Vec: 83% TF-IDF: 79%	CountVec: 93% Word2Vec: 89% TF-IDF: 76%	CountVec: 75% Word2Vec: 79% TF-IDF: 81%	CountVec: 87% Word2Vec: 89% TF-IDF: 90%

Figure 3: Performance on test set (After Oversampling)

In terms of model performances, in general, all models performed impressively in terms of Accuracy and Roc AUC scores. However, this is due to the imbalanced nature of the dataset. It is important for us to minimize the number of fraudulent listings classified as non-fraudulent by the model, as the cost of having such a wrong classification is high and impactful.

The XGBClassifier showed the best F1-Score performance when utilizing Word2Vec for text encoding. Conversely, for Precision and Recall metrics, the ExtraTrees Classifier and K Nearest Neighbors Classifier exhibited optimal results when employing Word2Vec embedding and TF-IDF for text encoding respectively.

Although we noticed a significant improvement in terms of Recall after we performed oversampling on the train set, it came at the cost of decreased Precision scores. Thus, the overall performance of models, especially in terms of F1-Score, did not increase for all models. This may be because some models, such as SVC, may be more sensitive to changes in the class distribution, causing them to be less robust when oversampling is performed.

4 Conclusion and Future Works

4.1 Conclusion

Our project successfully developed a machine learning-based fraud detection system, leveraging the Employment Scam Aegean Dataset (EMSCAD). This system represents a significant advance in the fight against online job fraud, offering a sophisticated tool for identifying fraudulent job postings. Through our comprehensive approach, encompassing data preprocessing, feature engineering, experimentation with various text vectorization techniques, and the implementation of diverse classifiers, we found that the XGB classifier with the Word2Vec technique performed most optimally, with an 87% F1-score.

Integration of Fraud Detection Model

With the creation of such a model, websites that help provide job listings could utilize it to filter possible fraudulent job listings. The model can be encapsulated within an API, which will be called once the platform receives a job posting. It would analyze the posting in real time and classify its validity before publishing the job posting. For job postings existing on the platform, a

batch-processing approach can be taken. This involves periodically running the model on all current listings to flag any potentially fraudulent posts that have been missed. Potential fraudulent postings identified can then be passed on to humans for manual evaluation if necessary. These new data can be fed back to the model as training data, continuously improving its adaptability to new fraud patterns.

Limitations

1. **Data Imbalance**
   The EMSCAD dataset's significant class imbalance presented challenges in model training and biased the system towards non-fraudulent postings. This could be shown in the trade-off between precision and recall we discovered when training the models.
2. **Temporal Scope**
   The dataset covers job listings from 2012 to 2014, potentially limiting the system's effectiveness against more recent fraudulent strategies.
3. **Limited Geographic Representation**
   The dataset's focus on specific regions may restrict the system's applicability across diverse markets and industries.

4.2 Future Works

1. **Dataset Enrichment**
   Future research should aim to incorporate more recent data, encompassing a wider variety of geographic locations, to enhance the system's relevance and accuracy in current and diverse market conditions.
2. **Deep Learning Approaches**
   Exploring deep learning models, such as more advanced neural networks, could potentially improve detection performances, particularly in processing complex text data and discerning subtle patterns indicative of fraud.
3. **Integration of a Unified Fraud Detection System across Job Platforms**
   Online job portals could partner with each other to integrate a unified fraud detection system to avoid platform-specific attacks. They could also share their data across platforms to collaboratively improve the system's capabilities.

5 References

[1] Liu, J. (2022, June 10). *Americans lost $68 million to job scams this year-here's what to look out for*. CNBC. https://www.cnbc.com/2022/06/10/americans-lost-68-million-to-job-scams-this-year-here-are-the-most-common-ones.html

6 Github Repository

https://github.com/LordZhiHao/BT4012_Fraud_Analytics_Project