Fake Job Post Detection Using Machine Learning

Final Report
ISSS610 Applied Machine Learning
Master of IT in Business
Singapore Management University

Group 5:
FELICIA ANTO CHRISTY
JOYCE WOON SHI HUI
NIKITHA BANDA
PANKAJ BHOOTRA
TAN MENGYUAN

Table of Contents

1 Business Problem and Use Case

Growing problem for job posting companies has been the fake job postings. In the recent times due to the increase in unemployment because of the pandemic, there has been an increase in such postings online[1]. A recent study conducted in America shows that there has been 70% increase in the number of fake job postings from the month of March to October in 2020[2] due to which more and more companies started to issue warnings about fake job offers[3]. Fake job postings not only collect personal information about an individual but also sometimes results in financial loss of that individual[4].

The aim of this project is to build a machine learning model that can be used by platforms such as Indeed and Monster to filter out the fake posting thereby enhancing user experience for both the candidate who is applying and recruiters who post the job descriptions on these platforms.

2 Dataset

The dataset used to build the models is "Employment Scam Aegean Dataset" (EMSCAD). It consists of 17880 real life job postings out of which 866 are fake job postings. The table below explains the 18 variables present in the dataset. The highlighted variable "fraudulent" is the target variable.

Data Source: http://emscad.samos.aegean.gr/

Column	**Description**	**Type**
title	Title of job	str
location	Geographical location of the job	str
department	Corporate department	str
salary_range	Indicative salary range	str
company_profile	Brief company description	html
description	Description of the job ad	html
requirements	Requirements of the job	html
benefits	Benefits offered by job	html

[1] Carmen R.(2020). Job scams have increased as Covid-19 put millions of Americans out of work. Here's how to avoid one. Retrieved from: https://www.cnbc.com/2020/10/06/job-scams-have-increased-during-the-covid-19-crisis-how-to-one.html

[2] Ivor B.(2020). Coronavirus: Thousands of jobseekers scammed in surge of fake employment listings. Retrieved from: https://news.sky.com/story/coronavirus-thousands-of-jobseekers-scammed-in-surge-of-fake-employment-listings-12117743

[3] BusinessToday.in. (2021). Beware of fake job offers: IndiGo issues advisory. Retrieved from: https://www.businesstoday.in/sectors/aviation/beware-of-fake-job-offers-indigo-issues-advisory/story/430150.html

[4] Casey C.(2020). Fake Jobs: Cybercriminals Prey on Job Seekers via Fake Job Postings. Retrieved from: https://securityboulevard.com/2020/01/fake-jobs-cybercriminals-prey-on-job-seekers-via-fake-job-postings/

telecommuting	t/f if telecommuting position	bin
has_company_logo	t/f for company logo present	bin
has_questions	t/f for screening questions present	bin
employment_type	Full-time, part-time, etc.	nom
required_experience	Experience required	nom
required_education	Education level required	nom
industry	Industry of job	nom
function	Function of job	nom
fraudulent	t/f for true or fake	bin
in_balanced	t/f for selected for balanced data	bin

Table 1: Data Description

3 Exploratory Data Analysis

EDA was carried out for categorical features, which are *required_education*, *required_experience*, *emplyment_type*, *function*, *industry*, and *location*. Indicator considered as part of the EDA is proportion, i.e., the number of specific type to the number of all types in overall setting, real jobs, and fake jobs. Compare column measure the proportion of specific type in fake jobs to that in real jobs, the calculation formula is (Proportion of Fake Job – Proportion of Real Job) / Proportion of Real Job.

3.1 *required_education*

About 40% fake jobs' *required_education* lie in high school level, while more than 50% true job *required_education* is bachelor's degree. Real jobs' education requirement generally higher than fake jobs'. Real jobs have vocational *required_education*, no fake job is of this type.

Type	Overall	Real Job	Fake Job	Compare
Associate Degree	2.80%	2.86%	1.45%	-49.30%
Bachelor's Degree	52.63%	53.90%	24.10%	-55.29%
Certification	1.74%	1.61%	4.58%	184.47%
Doctorate	0.27%	0.27%	0.24%	-11.11%
High School or equivalent	21.28%	20.41%	40.96%	100.69%
Master's Degree	4.26%	4.11%	7.47%	81.75%
Professional	0.76%	0.75%	0.96%	28.00%
Some College Coursework Completed	1.04%	1.06%	0.72%	-32.08%
Some High School Coursework	0.28%	0.07%	4.82%	6785.71%
Unspecified	14.29%	14.27%	14.70%	3.01%
Vocational	0.50%	0.52%	0.00%	-100.00%
Vocational - Degree	0.06%	0.06%	0.00%	-100.00%
Vocational - HS	0.09%	0.10%	0.00%	-100.00%

Table 2: Proportion of Different required_education Types in Overall, Real Job and Fake Job

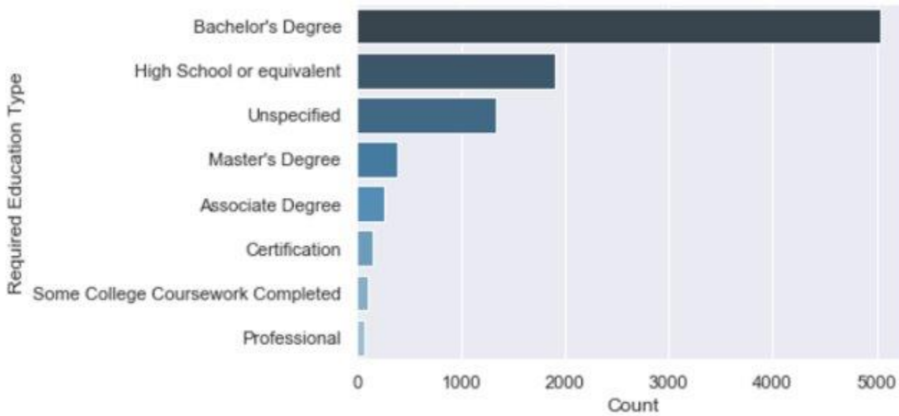

Figure 1: Distribution of True job postings under required_education

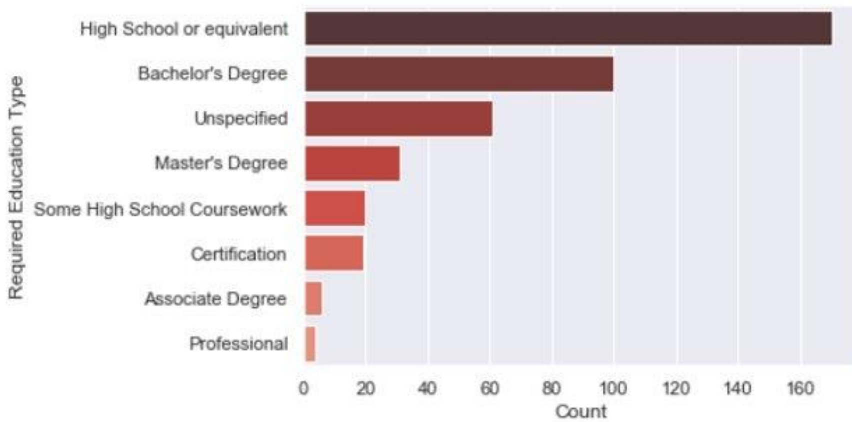

Figure 2: Distribution of Fake job postings under required_education

3.2 *required_experience*

More than 40% fake jobs are entry level, which is about 70% more than real jobs; There are very few executive level jobs, real or fake, but fake jobs claim to be executive levels more than real jobs; The largest percentage of real jobs is mid-senior level jobs, which account for more than 35% of all real jobs; Associate account for 21% of real jobs, but only 9% of fake jobs; Fake jobs have high probability of indicating 'Not Applicable'; Fake jobs have lower probability of being internship.

Type	Overall	Real Job	Fake Job	Compare
Associate	21.21%	21.68%	9.74%	-55.07%
Director	3.59%	3.58%	3.94%	10.06%
Entry level	24.90%	24.21%	41.53%	71.54%
Executive	1.30%	1.26%	2.32%	84.13%
Internship	3.52%	3.57%	2.32%	-35.01%
Mid-Senior	35.17%	35.54%	26.22%	-26.22%

level				
Not Applicable	10.30%	10.15%	13.92%	37.14%

Table 3: Proportion of Different required_experience Types in Overall, Real Job and Fake Job

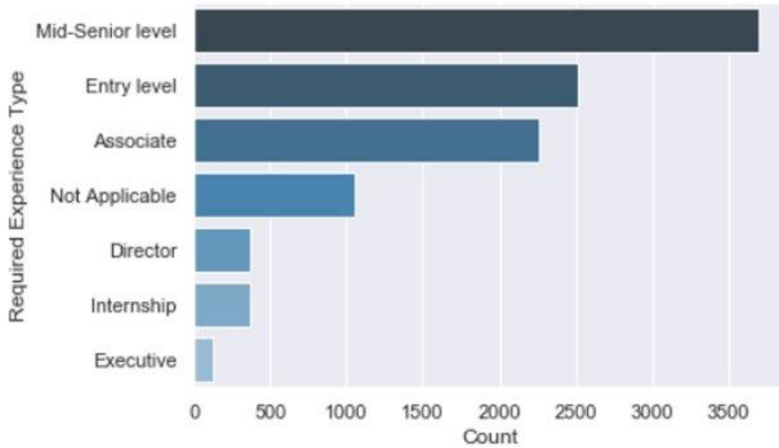

Figure 3: Distribution of True job postings under required_experience

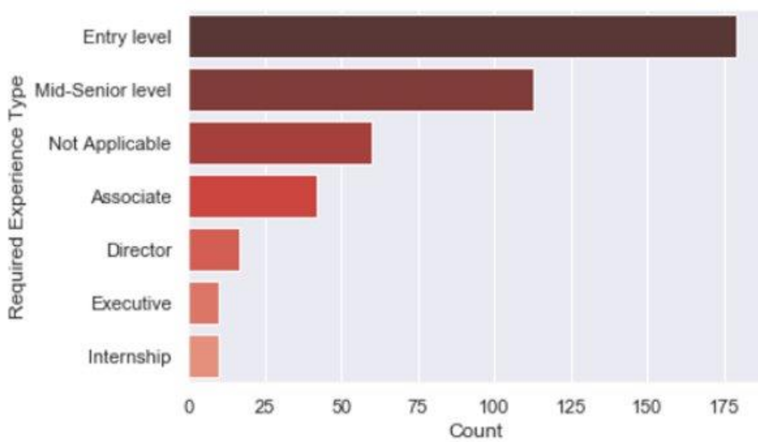

Figure 4: Distribution of Fake job postings under required_ experience

3.3 *employment_type*

Both real jobs and fake jobs mainly indicate full-time. Fake jobs have higher probability of being part-time job. Fake jobs sidestep being temporary or contract.

Type	Overall	Real Job	Fake Job	Compare
Contract	10.58%	10.74%	7.04%	-34.45%
Full-time	80.64%	80.75%	78.40%	-2.91%
Other	1.58%	1.54%	2.40%	55.84%
Part-time	5.53%	5.25%	11.84%	125.52%
Temporary	1.67%	1.73%	0.32%	-81.50%

Table 4: Proportion of Different emplyment_type Types in Overall, Real Job and Fake Job

3.4 *function*

Fake jobs are concentrated on Administrative and Engineering, accounting for 22% and 21% respectively; Distribution of real jobs is more diverse, with Information Technology account for 15%, followed by Sales (13%), Engineering (11%) and Customer Service (10%).

Type	Overall	Real Job	Fake Job	Compare
Information Technology	15.31%	15.76%	6.05%	-61.61%
Sales	12.85%	13.10%	7.75%	-40.84%
Engineering	11.80%	11.33%	21.36%	88.53%
Customer Service	10.76%	10.66%	12.67%	18.86%
Marketing	7.26%	7.53%	1.89%	-74.90%

Table 5: Top 5 Functions in Real Job and Proportion

Type	Overall	Real Job	Fake Job	Compare
Administrative	5.51%	4.69%	22.50%	379.74%
Engineering	11.80%	11.33%	21.36%	88.53%
Customer Service	10.76%	10.66%	12.67%	18.86%
Sales	12.85%	13.10%	7.75%	-40.84%
Information Technology	15.31%	15.76%	6.05%	-61.61%

Table 6: Top 5 Functions in Fake Job and Proportion

3.5 *industry*

18% of fake jobs belong to Oil & Energy, followed by Accounting (9%), Hospital & Health Care (8%) and Marketing and Advertising (7%); Information Technology and Services accounts for largest portion of real jobs (13%), followed by Computer Software (11%), Internet (8%) and Education Management (6%).

Type	Overall	Real Job	Fake Job	Compare
Information Technology and Services	13.36%	13.74%	5.41%	-60.63%
Computer Software	10.60%	11.07%	0.85%	-92.32%
Internet	8.18%	8.57%	0.00%	-100.00%
Education Management	6.33%	6.64%	0.00%	-100.00%
Marketing and Advertising	6.38%	6.32%	7.61%	20.41%

Table 7: Top 5 Industry in Real Job and Proportion

Type	Overall	Real Job	Fake Job	Compare
Oil & Energy	2.21%	1.44%	18.44%	1180.56%
Accounting	1.23%	0.82%	9.64%	1075.61%
Hospital & Health Care	3.83%	3.60%	8.63%	139.72%
Marketing and Advertising	6.38%	6.32%	7.61%	20.41%
Financial Services	6.00%	6.01%	5.92%	-1.50%

Table 8: Top 5 Industry in Fake Job and Proportion

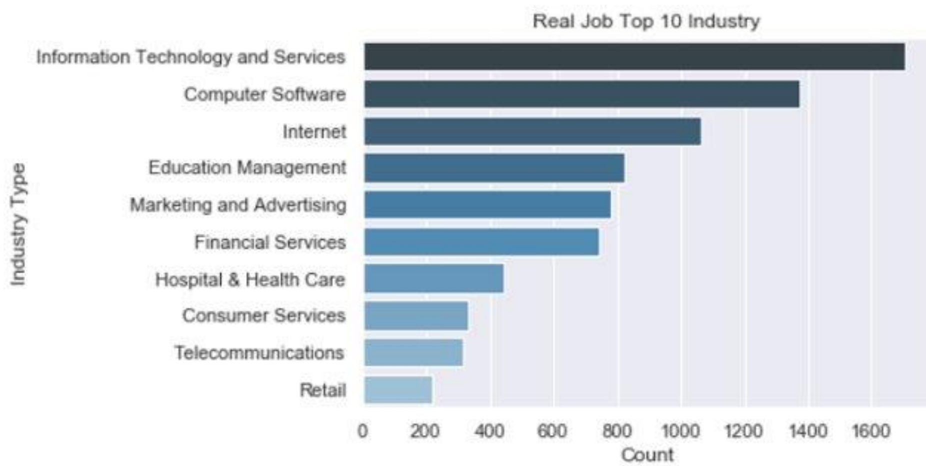

Figure 5: Distribution of True job postings under industry

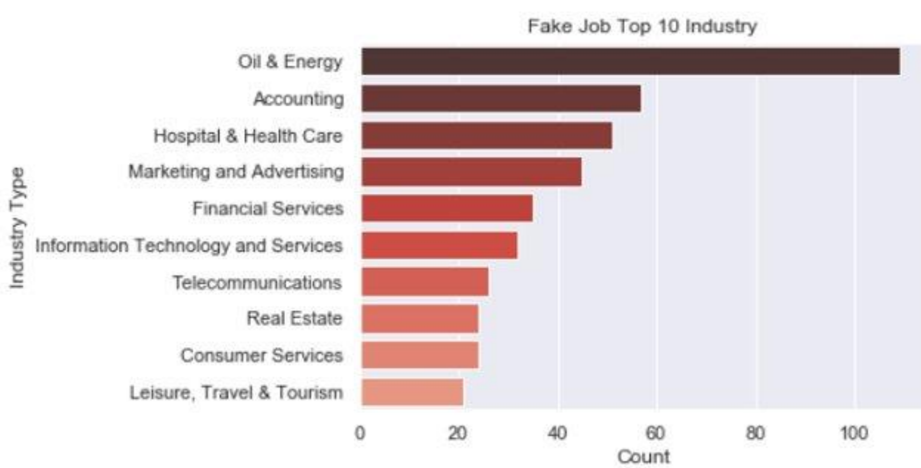

Figure 6: Distribution of Fake job postings under industry

3.6 *locations*

Most records are from the US. Real jobs mostly do not specify state or are from California. Whereas fake jobs are mainly from Texas.

States	Overall	Real Job	Fake Job	Compare
nil	14.43%	14.67%	9.70%	-33.88%
CA	11.47%	11.21%	16.51%	47.28%
NY	7.04%	7.00%	7.85%	12.14%
LND	5.55%	5.80%	0.69%	-88.10%
TX	5.45%	4.84%	17.55%	262.60%

Table 9: Top 5 States in Real Job and Proportion

States	Overall	Real Job	Fake Job	Compare
TX	5.45%	4.84%	17.55%	262.60%
CA	11.47%	11.21%	16.51%	47.28%
nil	14.43%	14.67%	9.70%	-33.88%
NY	7.04%	7.00%	7.85%	12.14%
MD	0.61%	0.43%	4.04%	839.53%

Table 10: Top 5 States in Fake Job and Proportion

4 Feature Engineering

Feature engineering was performed on the raw data before any data pre-processing steps were taken. This was due to the nature of the features engineered, which were largely based on the number of missing records and other irregularities in the data.

Firstly, for each row of data, the number of missing column values was computed and used as a new feature. Next, for each text-based column, the following were computed and included as new features:

- Word count
- Average word length
- Number of special characters (! @ # $ % & * ?)
- Number of digits
- Number of uppercase words

5 Data Pre-processing

5.1 Missing Values

There were several columns with missing values in the dataset. The table below shows the percentage of missing values from each of these columns.

Column	Percentage of missing values
salary_range	83.96%

department	64.58%
required_education	45.33%
benefits	40.25%
required_experience	39.43%
function	36.10%
industry	27.42%
employment_type	19.41%
company_profile	18.50%
requirements	15.04%
location	1.94%

Table 11: Percentage of missing values per column

Out of these, *department* and *salary_range* were removed from the dataset, as they have a significantly high proportion of missing values at 64.58% and 83.96% respectively.

5.2 Categorical Variables

There was a total of 9 columns which were categorical in nature. As shown in the table below, out of these categorical variables, 3 were binary and the rest had a range of 9 to 132 unique categories.

Column	**Number of unique categories**
telecommuting	2
has_company_logo	2
has_questions	2
employment_type	6
required_experience	8
required_education	14
function	38
industry	132

Table 12: Number of unique categories per column

For variables with less than 30 unique categories, namely *telecommuting*, *has_company_logo*, *has_questions*, *employment_type*, *required_experience* and *required_education*, one-hot encoding was performed.

For *function* and *industry*, which have many unique categories, the number of unique categories was reduced to 30. To do so, the top 29 categories with the highest frequencies were retained. The remaining categories were combined into one merged category. After which, one-hot encoding was performed on these columns.

5.3 Text-based Variables

There were 6 text-based columns in the dataset, namely *title*, *location*, *company_profile*, *description*, *requirements* and *benefits*. Some of these columns were HTML-formatted.

All the text-based columns were first cleaned with the following steps:

- Conversion from HTML to text format
- Conversion to lowercase
- Removal of HTTP/URLs
- Removal of special characters
- Removal of numbers
- Removal of non-English words
- Removal of stop words

Subsequently, each text-based column was vectorized using a TF-IDF vectorizer. Column-wise vectorization was performed in order to retain the differences in meaning and significance of similar words across different columns.

Truncated SVD was employed to reduce the dimensions of the resulting vectors from the column-wise vectorization. The number of components retained per column corresponds to the ratio of unique words originally found in each column. As a result, a total of 100 word vectors were chosen from all the text-based columns combined.

6 Models and Results

6.1 Overview

Next, we are going to cover the Machine Learning models that we trained on the preprocessed dataset, along with the detailed analyses and results for each. Before we deep dive into each individual model, we first want to highlight some aspects of model analysis which were common across all algorithms:

1. **Splitting and Scaling:** Our source dataset is highly imbalanced with only 866 samples of fake jobs while around 17,000 are real ones. We maintain a stratified split between training set (80%) and testing set (20%) to maintain this percentage distribution and ensure that testing set does not end up with only real job samples. This testing set is isolated and plays no part in model training or optimization to keep it 'unseen'. Also, each of the 220 features computed from preprocessing are normalized to a value between 0 and 1 using Min-Max Scaler, to ensure that all features have similar ranges and that the features with numerically larger values do not have higher intrinsic influence on the outcome, especially for features computed with TF-IDF.

2. **Data Sampling Approaches:** Due to data imbalance, we have tested each Machine Learning algorithm with two versions of the preprocessed dataset. One maintains the original preprocessed set with very few fake jobs. In the other, we are doing an up-sampling of fake jobs using a synthetic resampling method named SMOTE. SMOTE creates new training samples of the minority class by combining a subset of examples from the original data that lie close to each other in the feature space. The training samples are normalized/rescaled with Min-Max Scaler before applying SMOTE. Testing set is not up-sampled to maintain its originality. As we would see later, there are some models that perform better on the original dataset while some performed better on the re-sampled dataset.

3. **Cross Validation Analysis:** To ensure that our models do not overfit a fixed validation set, we ran a 5-fold cross validation on every model skeleton, using scikit-learn's StratifiedKFold API. This ensures that each validation fold is a stratified split from the training fold. For every combination of training and validation fold, we also ensure that only the training set is up-sampled with SMOTE and that the validation set is left as original, otherwise it would skew the results of our evaluation metrics. For every fold, we calculate the F-2 score instead of accuracy (more on this in next point) and take the average for all 5 folds at the end as our main result. Detailed code, with the example of Logistic Regression, is as follows:

```python
from sklearn.model_selection import StratifiedKFold
from imblearn.over_sampling import SMOTE

skf = StratifiedKFold(n_splits=5, shuffle=True)
fbeta_scores = []

for fold, (train_index, test_index) in enumerate(skf.split(x_train, y_train), 1):
    x_fold = x_train[train_index]
    y_fold = y_train[train_index]
    x_val = x_train[test_index]
    y_val = y_train[test_index]
    sm = SMOTE(random_state = 5)
    x_upsampled_fold, y_upsampled_fold = sm.fit_resample(x_fold, y_fold)
    model = linear_model.LogisticRegression(C=1, penalty='l1', solver = 'liblinear', random_state=1)
    model.fit(x_upsampled_fold, y_upsampled_fold)
    y_pred = model.predict(x_val)
    fbeta = metrics.fbeta_score(y_val, y_pred, beta=2)
    print(f'For fold {fold}:')
    print(f'F-2 score: {fbeta}')
    fbeta_scores.append(fbeta)

print(sum(fbeta_scores) / len(fbeta_scores))
```

Figure 7: Code for cross validation

4. **Evaluation Metrics:** We are looking at Recall, Precision, F-2 Score and ROC-AUC scores. Our dataset is highly imbalanced, because of which accuracy is not the right metric to use. Our main challenge is to ensure that we classify our minority class (fake jobs) as correctly as possible. Hence, in this binary classification, we mark the fake job samples as the positive class and optimize for recall over precision. This is also the rationale for choosing F-2 score instead of F-1.

5. **Hyperparameter Tuning and Other Analysis:** Every model examined had different hyperparameters with various possible values. To ensure that we configure the model without overfitting, we examined the difference between training set error and validation set error for different hyperparameter settings and attempted to maintain consistent error across both. Where feasible, grid search is also conducted on different hyperparameter values. Apart from capturing the evaluation metrics along this journey, we also analyzed the top features determined from every model to check its consistency with our EDA.

   Broadly, we have looked at three categories of algorithms as follows:
   1. **Simple Classifiers:** Logistic Regression, Support Vector Classifier
   2. **Ensemble Methods:** Random Forest, XG Boost
   3. **Outlier Detection:** One-Class SVM

6.2 Simple Model: Logistic Regression

We started by training a naïve Logistic Regression model on the original preprocessed dataset (without upsampling), with default hyperparameter values and obtained a 5-fold cross-validation score of 45.7% (F-2 score). We trained this same model on the upsampled version of the dataset (with SMOTE) and the cross-validation score jumped to 65.6% (the validation set as mentioned earlier is not upsampled). This gave a clear indication that further exploration on Logistic Regression model should be continued with the upsampled dataset.

Our next objective was to find the right hyperparameter settings to fit our upsampled dataset. One of the main hyperparameters here is the inverse regularization strength (C). To find the ideal value for C that minimizes training set and validation set errors without underfitting/overfitting, we calculated the inverse log likelihood on both training set and validation set for different values of C, namely 0.001, 0.01, 0.1, 1, 10, 100, 1000. These error values are then plotted as follows (blue=training, red=validation):

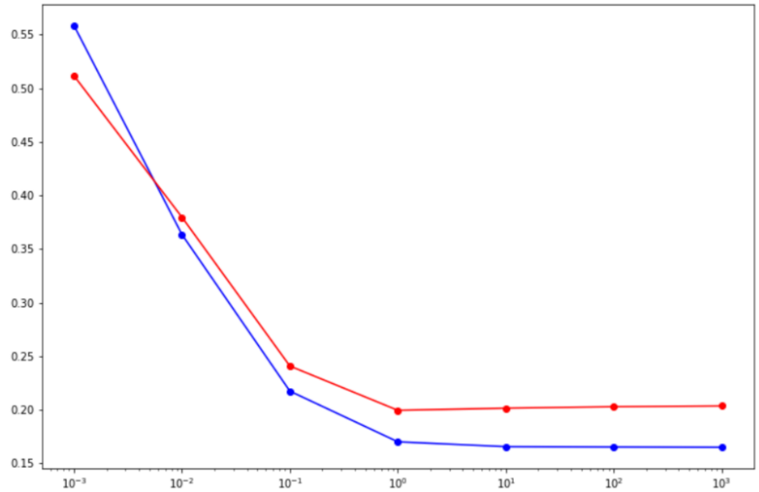

Figure 8: Change in training set and validation set errors for different values of C

This confirmed that for C = 1, both training and validation set errors are minimized and that for higher values of C, the performance gets worse for validation set, showing signs of overfitting. We also found that setting penalty = ‘l1’ gives a slight improvement instead of default penalty (‘l2’), which makes sense as we have 220 features in our preprocessed dataset, some of which may not be relevant and L1/Lasso penalty automatically eliminates such features by setting weights to 0. The final cross-validation score slightly improves to **67.2%** and scores on the testing set were obtained as follows:

	Imbalanced Train	**Upsampled Train**
Precision	86.24%	36.30%
Recall	54.34%	89.00%
F-2	58.68%	69.00%
ROC-AUC	0.963	0.971

Confusion Matrix	TN	3388	FP	15	TN	3133	FP	270
	FN	79	TP	94	FN	19	TP	154
Top 5 Features	US as *location* Word count of *location* Average word length of *company_profile* Word “secure” in *company_profile* Word “new” in *description*				Number of special characters in *description* US as *location* Average word length of *company_profile* Word “secure” in *company_profile* Word count of *location*			

Table 13: Logistic regression optimal model scores on testing set for imbalanced training set vs. upsampled training set

Some of the eliminated features from the best model (using Lasso) are as follows:

Some Eliminated Features
Average word length of *location*
Master’s Degree as *required_education*
Administrative as *function*
employment_type
Word count of *description*

Table 14: Eliminated features

6.3 Simple Model: Support Vector Classifier

Initially, a support vector classifier model was built using the original preprocessed dataset (without upsampling) with default kernel=“linear” and an optimal C value at C=100 with which F2 score of 64.93% was attained. When the same model was built using the upsampled version of the dataset (with SMOTE), it gave F2 score of 68.7%. With the improvement in result, we moved forward in using the upsampled version of the dataset to build the SVC models. Below is the table of various parameters and their respective values that were tuned.

Parameters	Values
kernel	{ “linear”, “poly”, “rbf” , “sigmoid”}
gamma	{0.001, 0.01, 0.1, 1, 10} (not for “linear”)

Table 15: SVC parameter setting

Based on all the models built by tunning the parameters shown above, the best performing model was the one where kernel=”poly” and gamma=0.1 with an F2 score of 82.6% on the testing data. The table below shows the other scores of this model along with the optimal model built on unsampled dataset.

	Imbalanced Train	Upsampled Train
Precision	81.06%	68.02%
Recall	61.85%	87.28%
F-2	64.93%	82.6%
ROC-AUC	0.953	0.98

Confusion Matrix	TN	3378	FP	25	TN	3332	FP	71
	FN	66	TP	107	FN	22	TP	151
Top 5 Features	Number of digits in description Word "safe" in *company_profile* Average word length of location Number of special characters in *company_profile* Number of special characters in requirement				(since the kernel is “poly” the weights of features are not accessible to identify the important features)			

Table 16: SVC optimal models scores

With the optimal parameters of kernel=”poly” and gamma=0.1, we obtained a 5-fold cross-validation score of **78.42%** by taking the average of the 5 folds. This shows that there is no overfitting in the models. It is also notable that with upsampling and training the model, there has been an increase in recall resulting in increase in the F2 score.

6.4 Ensemble Model: Random Forest Classifier

The Random Forest Classifier fits many decision tree classifiers on various sub-samples of the data and performs averaging to increase the predictivity.

We built Random Forest Classifier models using both the unsampled data and the upsampled data and found the model with the upsampled data to yield better results. Following this, we tried hyperparameter tuning, but resorted to using the default parameters as they yielded the best results.

The following table shows the results of the 2 best performing models, using the imbalanced and upsampled datasets.

	Imbalanced Train				**Upsampled Train**			
Precision	99.00%				87.18%			
Recall	57.23%				78.61%			
F-2	62.50%				80.19%			
ROC-AUC	0.986				0.989			
Confusion Matrix	TN	3402	FP	1	TN	3383	FP	20
	FN	74	TP	99	FN	37	TP	136
Top 5 Features	Average word length in *company_profile* Word "safe" in *company_profile* Word "abroad" in *company_profile* Word "secure" in *company_profile* Word "get" in *company_profile*				Has *company_logo* Average word length in *company_profile* Word "get" in *company_profile* US as location Word "safe" in *company_profile*			

Table 17: Random Forest optimal models scores

The model trained on the upsampled dataset yielded a 5-fold cross validation score of **73.40%.**

Between these 2 models, the model trained on the upsampled dataset gave between recall, F2-score and ROC-AUC even though it compromised a little on precision. However, the improvement in recall and subsequently the F2-score exceeded the drop in precision, making the latter model better suited for our problem. As from the top features identified, we can see that generally features based on the company profile, such as the average word length, and the presence of words such as “safe” in the company profile have contributed to the model's prediction. This highlights the importance of the company profile in deciphering if a job post is fake or real.

6.5 Ensemble Model: XGboost Classifier

XGBoost is also know as eXtreme Gradient Boosting and it is an ensemble method that works by boosting trees using a gradient descent algorithm. It corrects previous mistakes made by the model, learn from the mistakes and improves the next step until there is no scope of further improvement. It is popular as it is fast and gives good accuracy.

For this project, we fitted XGBoost model on both imbalanced train and upsampled train and performed hyperparameter tuning on the upsampled train model. Below is the table of various parameters and their respective values that were tuned.

Parameters	Values	Default Value	Final Value
min_childe_weight	{ 1, 3, 5, 10}	1	1
gamma	{ 0, 2, 5, 10 }	0	0
subsample	{ 0.5, 1, 5, 10 }	1	0.5
colsample_bytree	{ 0.5, 0.8, 0.9, 1 }	1	1
max_depth	{ 5, 6, 8, 9, 10 }	6	10

Table 18: XGboost parameter setting

The model scores are as follows:

	Imbalanced Train Default Parameters				**Upsampled Train Tuned Parameters**			
Precision	94.12%				87.65%			
Recall	73.99%				82.08%			
F-2	77.30%				83.14%			
ROC-AUC	0.8688				0.9075			
Confusion Matrix	TN	3395	FP	8	TN	3383	FP	20
	FN	45	TP	128	FN	31	TP	142
Top 5 Features	Word "get" in *company_profile* Word "safe" in *company_profile* Average word length of *company_profile* US as location				Has *company_logo* Word "get" in *company_profile* US as location Word "safe" in *company_profile* Average			

	Has *company_logo*	word length of *company_profile*

Table 19: XGboost optimal model scores

For “Upsampled Train Tuned Parameters”, we obtained a 5-fold cross-validation score of **79.98%**.

We note that between imbalanced train and upsampled train, while there was a slight decrease in precision, there was an increase in recall, F-2 and ROC-AUC, which is the improvement we were seeking. However, we also note that the changes between imbalanced train and upsampled train was not significant (changes all occurred within 10%). There were even marginal changes in Top 5 features where all Top 5 features are the same, except in different order. These slight changes between imbalanced and upsampled train might be due to XGBoost’s capability in handling imbalanced data.

6.6 Outlier Detection: One-Class SVM

In this part of exploration, we treat fake job postings as outliers. Rationality comes from the ratio of fake job to real job which is only 0.051. Since the dataset is imbalanced with very less records of fake posts, we consider this class to be the outliers in One-Class SVM model.

Unlike the previous models, we use the unsampled dataset and the difference that lies in One-Class SVM is that only inliers of training data will be used to fit the model.

One-Class SVM does not have predict_proba method, thus decision_function was used to obtain the signed distance between the hyperplane and all instances in test set and the threshold was adjusted to maximize F-2 score. With default parameter values, an F2 score of 25.45% was attained. Therefore to further improve the score, various parameters and their respective values that were tuned as showed in the table below.

Parameters	Values
kernel	{“sigmoid”, “linear”, “rbf”, “poly”}
gamma	{“auto”, “scale”} (not for “linear”)
nu	{# of fake jobs/# of real jobs, 0.5, 0.6, 0.7, 0.8, 0.9}

Table 20: One-class SVM parameter setting

Based on all the models built by tunning the parameters shown above, the best performing model was the one where the parameters setting are {kernel:”sigmoid”, gamma:”scale”, nu:0.8}.

	Imbalanced Train Default Parameters				Imbalanced Train Tuned Parameters			
Precision	8.30%				10.81%			
Recall	52.60%				66.47%			
F-2	25.45%				32.74%			
ROC-AUC	0.6546				0.7458			
Confusion Matrix	TN	2398	FP	1005	TN	2454	FP	949

	FN	82	TP	91	FN	58	TP	115

Table 21: One class SVM optimal models scores

The performance of trained One-Class SVM model is not good on test set and this could be due to the selection of insignificant features. There are 220 features after pre-processing and the performance of model indicates that not all features are significant. From the figures below (yellow dots represent fake jobs, purple dots represent real jobs), it is noted that by adding one significant feature to the mixture of fake jobs and real jobs the separation of fake jobs and real jobs improved.

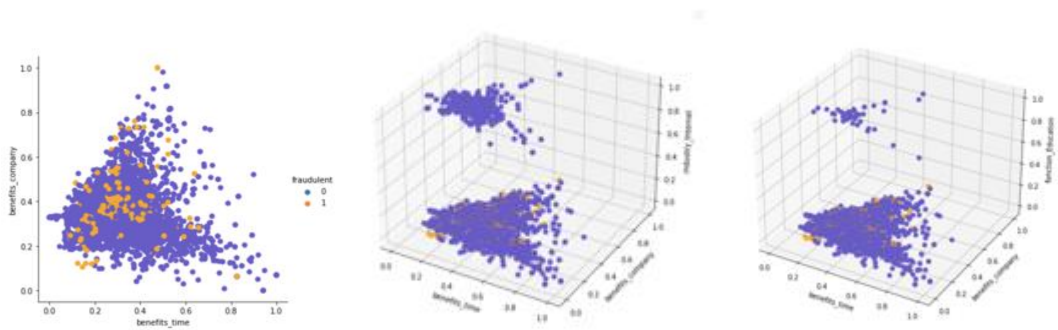

Figure 9: Add Significant Feature to Mixture of Fake Job and Real Job

Figure exploration is considered to be an optimal method to do feature exploration, but it's feasibility is limited when the feature dimension is high like in our project. To be more efficient, PCA was used to find the dominant features. From the table below, top 5 features of real job and top 5 features of fake job can explain 38.51% and 43.03% variance of corresponding set.

Top 5 Features of Real Job	Explained Var	Top 5 Features of Fake Job	Explained Var
Required experience	0.1710	*Required education*	0.2072
Employment type Full-time	0.0746	*Required experience Entry level*	0.0903
Required experience Entry level	0.0550	*industry*	0.0532
Has questions t	0.0505	*Location US*	0.0419
location us	0.0340	*Employment type*	0.0376
Total	**0.3851**	**Total**	**0.4303**

Table 22: Important features in One-Calss SVM model

Top 40 features of real jobs were used to fit into One-Class SVM and the model was measured by the same metrics as before. From the table below, we can see that the model reached ROC of 0.661 after parameter fine tuning (parameter setting are {kernel:"sigmoid", gamma:"scale", nu:0.6}) but still it is not as good as the previous model with ROC value of 0.7458.

	Imbalanced Train Default Parameters
Precision	10.15%

Recall	57.85%
F-2	29.58%
ROC-AUC	0.661

Table 23: One-Class SVM model scores after PCA

6.7 Summary of Results

The below table shows a summary of the optimal results of all the models built for this project.

Score Type	LR	SVC	RF	XGB
Precision	36.3%	68.02%	87.18%	87.7%
Recall	89.0%	87.28%	78.61%	82.1%
F-2	69.0%	82.6%	80.19%	**83.1%**
ROC AUC	0.971	0.98	0.989	0.907
5-Fold Cross Val (F-2)	67.2%	78.4%	73.40%	**79.0%**
Top 5 Features	Number of special characters in *description*	-	Has *company_logo*	Word "get" in *company_profile*
	US as *location*	-	Average word length in *company_profile*	Has *company_logo*
	Average word length of *company_profile*	-	Word "get" in *company_profile*	US as *location*
	Word "secure" in *company_profile*	-	US as *location*	Average word length of *company_profile*
	Word count of *location*	-	Word "safe" in *company_profile*	Word "safe" in *company_profile*

Table 24: Summary of optimal model scores

Generally, all the models trained were able to get good F2-scores, and ROC AUC above 0.9. Based on the 5-fold cross validation scores, we can see that the XGBoost Classifier worked best for this problem, with a F2-score of 79.0%.

As for important features contributing to the models' predictions, attributes related to the company profile were repeated between the different models. Particularly, the average word length and use of words such as "safe" and "secure" in the company profile seem to be important features in deciphering if a job post is fake or real. The location being "US" is also consistent across all models.

7 Future Improvements

We noted several limitations and made some suggestions to improve the analysis and models such as the dataset used for the model is not very varied in terms of representation. Amongst other parameters, it is skewed by location as most job listings are posted in the United States, which might affect the model's ability to predict fake jobs in other countries. To possibly counter this, perhaps sourcing for a more varied dataset in terms of country, industry and function, etc, should be performed. Possible sources for real job postings include platforms such as LinkedIn, Indeed, Monster or other job boards. However, it should be noted that it is a challenging effort to consolidate fake job posts which may require heavy consolidation and cleaning/preprocessing from multiple sources.

Many job boards have the concept of user profiles or company profiles, as entities must first sign in before they can post any job. This allows these platforms to track other external features about the job poster, such as their rating, browsing history, background, etc. This history can come in very handy when predicting for fake jobs. For instance, a multi-national company with a long history of job postings on a given platform is very likely to post only real jobs. At the same time, there could be entities who 'pose' as MNCs or representatives of MNCs, whose ingenuine profile would give away their job posts as fake. Hence, to improve on the current model, it would also be important to capture external features other than job description for this problem.

During our analysis of One-Class SVM, we found that this does not perform very well and one of the fundamental reasons here is that the outliers (fake jobs) are not easily distinguishable from the inliers (real jobs) when plotting them in the feature space. This showed that One-Class SVM requires even more effort on preprocessing to clearly distinguish between the classes in the vector space. To reach better results, features that can significantly distinguish fake jobs and real jobs are expected and the way to find significant features need further investigate. One-Class SVM model is quite different from casual machine learning models, in this case we can put some effort on the correctness usage of this model and do calibration.

Lastly, exploring deep learning approaches would help to improve model performance. We performed a very quick test with a dense neural network built with Keras as shown in the next image, and it gave comparable performance as XG-Boost (F-2 Score: ~82%).

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_30 (Dense)             (None, 128)               28288
_________________________________________________________________
dense_31 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_32 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_17 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_33 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_18 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_34 (Dense)             (None, 16)                528
_________________________________________________________________
dropout_19 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_35 (Dense)             (None, 2)                 34
=================================================================
Total params: 43,346
Trainable params: 43,346
Non-trainable params: 0
_________________________________________________________________
```

Figure 10: Dense Neural Network that we quickly built and tested with Keras

8 Conclusion and Takeaways

We have successfully achieved a viable solution for detecting fake jobs in a real world setting. Our best model is XG-Boost, with high F-2 Score of **83.1%**. We prioritize recall over precision and our F-2 Score shows that the model can correctly classify a significant portion of fake jobs as truly fake when deployed in a production setting.

Our model can be successfully employed in an actual business scenario as follows:

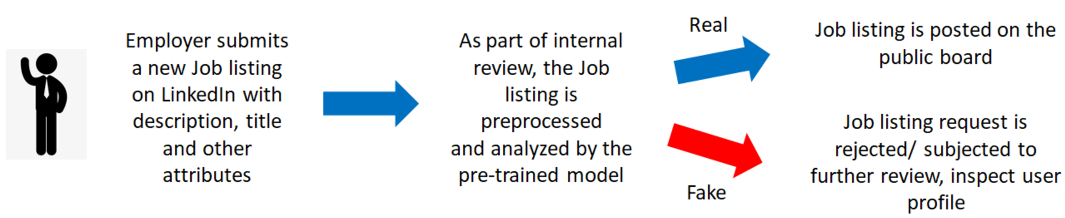

Figure 11: Demonstrating business workflow for detecting fake jobs and where our model fits in

One of our key takeaways from this experience is the impact of pre-processing and feature engineering on the solution outcome. We did a thorough EDA on our raw dataset as demonstrated earlier, which gave us insights on fake/real distributions across various attributes, be it country, industry, function, etc. Bearing some of our observations from EDA, we did some relevant feature engineering to record several data points such as word count of job description, number of special characters in benefits, etc. These engineered features turned out to be the main differentiators between fake and real jobs across all our models.

We hope to continue improving our solution by exploring other new approaches across data preprocessing, feature engineering, deep learning, etc.