Detecting Fraudulent Job Posts through Textual Features

By Vincent Kwan (301147654)

Advisors: Prof. Will Au, Prof. David Parent

Completed for: BA 723 (Capstone)

**Table of Contents**

[**Executive Summary** 4](#_Toc111976667)

[**Executive Objective** 4](#_Toc111976668)

[**Summary of Modelling** 4](#_Toc111976669)

[**Executive Findings** 5](#_Toc111976670)

[**Recommendations** 5](#_Toc111976671)

[**Introduction** 6](#_Toc111976672)

[**Problem Statement** 6](#_Toc111976673)

[**Objectives** 7](#_Toc111976674)

[**Assumptions regarding Dataset** 7](#_Toc111976675)

[**Project Dependencies** 7](#_Toc111976676)

[**Dataset Description** 8](#_Toc111976677)

[Limitations of the Dataset 8](#_Toc111976678)

[**Data Engineering** 10](#_Toc111976679)

[Engineered Variables 10](#_Toc111976680)

[**Text Cleaning & Processing (Lemmatization)** 10](#_Toc111976681)

[**Text/Post Characteristics Variables** 11](#_Toc111976682)

[Missing Indicators 11](#_Toc111976683)

[Imputation 12](#_Toc111976684)

[Sentiment Analysis 12](#_Toc111976685)

[Reading Score & Average number of words per sentence 13](#_Toc111976686)

[Collapsing/merging categories 14](#_Toc111976687)

[**Term-Document Matrix & TF-IDF** 16](#_Toc111976688)

[**Upsampling with SMOTE** 16](#_Toc111976689)

[**One-hot encoding** 16](#_Toc111976690)

[**Clickbait words & Clickbait Ratio** 16](#_Toc111976691)

[**Datasets Created** 17](#_Toc111976692)

[**Data Summary** 18](#_Toc111976693)

[**Modelling** 19](#_Toc111976694)

[**Baseline Results** 19](#_Toc111976695)

[**Modelling Approach** 19](#_Toc111976696)

[Model Inventory 20](#_Toc111976697)

[Evaluation Statistic 20](#_Toc111976698)

[**Naïve Bayes Classifier** 21](#_Toc111976699)

[Categorical Naïve Bayes 21](#_Toc111976700)

[Complement Naïve Bayes 24](#_Toc111976701)

[**Tree Models** 29](#_Toc111976702)

[Decision Trees 29](#_Toc111976703)

[Random Forests 34](#_Toc111976704)

[**Gradient Boosted Trees** 38](#_Toc111976705)

[Histogram Gradient Boosted Tree 39](#_Toc111976706)

[Light Gradient Boosting Machine 44](#_Toc111976707)

[**Neural networks** 48](#_Toc111976708)

[Variational Autoencoder 48](#_Toc111976709)

[**Champion Model** 54](#_Toc111976710)

[**Insights** 55](#_Toc111976711)

[**Governance & Control** 57](#_Toc111976712)

[**Risk Matrix** 57](#_Toc111976713)

[**Shifting Targets** 58](#_Toc111976714)

[**Dependency Creep** 58](#_Toc111976715)

[**Suspicious Results** 59](#_Toc111976716)

[**Missing Values & Imputation** 59](#_Toc111976717)

[**Noise in the dataset** 59](#_Toc111976718)

[**Controlling for Drift** 59](#_Toc111976719)

[**Recommendations** 61](#_Toc111976720)

[**Real-time Learning System** 61](#_Toc111976721)

[**Protection Guidelines** 61](#_Toc111976722)

[**Feedback Platform** 61](#_Toc111976723)

[**Job Ad Dataset** 61](#_Toc111976724)

[**Conclusion** 62](#_Toc111976725)

[**References** 63](#_Toc111976726)

# **Executive Summary**

This project was undertaken with the goal of analysing a dataset consisting of legitimate and fraudulent job advertisement posts. The goal was to seek out Multi-level Marketing posts, instances of corporate identity theft, and ‘stealth’ fraudulent posts (posts masquerading as legitimate offers). However, the end-goal was not to create a model that could discriminate between fraudulent and legitimate posts, but rather to develop a methodology to analyse textual features in job posts, and to derive insights from the data. The Employment Scam Aegean Dataset (EMSCAD) was used for the project – this dataset contains 17,880 real-life job ads collected in the period 2012-2014.

Fake job posts are designed to collect sensitive personal information from job-seekers, or outright scam them of their money (Edmonton Police, n.d.). These posts are predatory in nature, and target vulnerable and desperate sections of society. This project has special personal importance to me as I have seen many fraudulent jobs posts myself.

A novel way of tackling this problem was undertaken – creating a Variational Autoencoder with a compressed encoder in order to learn the latent space distribution of the input features. The learnt distribution was used by the decoder to reconstruct the input features – and the reconstruction loss was used to determine whether the particular job post was fraudulent or legitimate.

I was able to find several indicators of fraud in a job post – most of these indicators were derived from post/text characteristic features (e.g., number of whitespaces used, consecutive punctuation marks, etc). The main finding was that most fraudulent job posts target lower-educated jobs/professions, including clerk/secretary roles. Fake jobs also prioritize cash rewards/bonuses/perks in their profiles and descriptions as opposed to legitimate job posts prioritizing job security, reputation, challenge, and prestige.

## **Executive Objective**

Given a dataset consisting of the details of job postings on a website, detect whether that particular job posting is fraudulent. This involves seeking out multi-level Marketing postings, instances of Corporate Identity fraud, and instances of ‘stealth’ fraudulent posts.

## **Summary of Modelling**

Fifteen models were created in total consisting of two variants of each major class of model. The models consisted of Naïve Bayes Classifiers (Categorical & Complement), Decision Trees & Random Forest Classifiers (fitted with Cross-Validation), Gradient Boosted Trees (Sci-kit Learn’s implementation & LGBM), and finally the Variational Autoencoder.

Extensive data cleansing and preparation was carried out in order to make the data compatible with the models. Feature Engineering was also carried out – creating new features from the text variables. A term-document matrix, and associated TF-IDF scores were created from the text features – this was used with the VAE for classification.

## **Executive Findings**

1. Text/Post characteristics such as the amount of whitespace used in the post, the average number of words per sentence, consecutive punctuation marks, etc. may provide enough predictive power to distinguish between fraudulent & legitimate posts.
2. Out of the text/post characteristics: the amount of whitespace in a post, clickbait ratio (clickbait defined in feature engineering), and average amount of words per sentence are the most important features.
3. Scammers disproportionately target vulnerable and desperate people. They design their posts with lower education & work experience requirements, as well as extraordinarily high compensation for the role. Furthermore, they focus on instant cash rewards & bonuses.

## **Recommendations**

Based on my findings, I recommend:

1. **Implementing a Continuous Learning system:** The fraud detection problem is especially suited for real-time analytics where statistics and insights can be generated ‘on-the-fly’. Furthermore, a live-model which continuously updates its weights and learning parameters from new data would be highly suited for this problem.
2. **Protect Vulnerable Users:** Based on my findings, fraudsters target vulnerable and desperate people. They frequently highlight cash rewards, and easy compensation as remuneration. Posts targeting these users, and having similar rewards should be put under greater scrutiny. Potentially suspect posts should be flagged, and users should be informed – similar to social media websites placing a disclaimer on political posts, or posts about COVID-19.
3. **Create a platform for Users:** Increasingly, discussion forums and Report features have been given less importance by job websites and job boards. These features should be given greater importance in order to give power back to the users. These platforms will allow users to share personal experiences regarding suspicious job advertisements – peer policing will help ensure a higher quality of job posts, and prevent scams.
4. **Create a job frauds dataset:** The problem of fraudulent job posts is of great importance, considering the monetary and time resources wasted by scammers. Therefore, in order to better study and combat this issue – a job frauds dataset should be created. This dataset would be akin to the ImageNet dataset for image classification. The dataset will allow data scientists, analysts, and programmers to work together in finding significant features to root out fake job ads. Furthermore, this dataset should be continuously updated – allowing users to add suspect posts. This task is particularly labor-intensive, and would require thousands of collaborators to create a labelled dataset.

# **Introduction**

Have you ever gone through a job advertisement – only to find it too good to be true? Those ads are fake job posts. These posts are intentionally designed to amaze and shock – the remuneration and compensation packages in these posts are frequently higher than average for similar posts, and work/educational requirements are often lower . These posts are also designed to attract the vulnerable and desperate sections of society – offering instant cash bonuses, and generous work-hours. Fraudulent job posts masquerade as legitimate job offers with an aim to extract personal & sensitive information from job-seekers, and to scam them of their money. While fake job ads are generally just a nuisance to be ignored for most people, their harmful effects to society cannot be ignored.

In 2019, there were an estimated 14 million victims of job fraud with $2 billion in direct losses. During 2020, complaints to the Canadian Anti-Fraud Centre regarding job fraud doubled (CBC News, 2021). The rise in popularity of online work has led to an implosion of fraudulent job posts. Job candidates are finding it increasingly difficult to distinguish between fraudulent and legitimate job posts due to not being able to physically verify (as per the online/WFH model). Scammers are taking advantage of the Work-from-home model to dupe potential candidates and job-seekers. They pose as legitimate businesses, and even go so far as to create fake emails & websites.

Fake job postings on job portals leads to a poor customer experience for job-seekers and students. But the damage extends beyond a poor experience – people scammed by fake jobs lose time and money. While most discerning people don’t fall for the scams, some people still fall prey. According to the Better Business Bureau (2022), the median financial loss due to fake job scams was $500 (CBC News, 2021). Young professionals in the age group of 25 to 35 years old were victimized heavily, and women comprised 62% of the victims. In a report released by the Federal Trade Commission (FTC) – in USA, Americans lost $68 million to job scams in 2022 alone (Liu, 2022).

## **Problem Statement**

The problem consists of identifying fraudulent posts in a highly imbalanced dataset. This involves developing models, insights & heuristics in order to distinguish between fraudulent and legitimate job posts. This task is not easy, as fraud detection consists of trying to aim at a moving target. Scammers change tactics and strategy over time in order to avoid detection, and empirical rules to detect scams developed in the past may no longer be effective in the present/future. That said, there are certain stereotypes/tropes surrounding fake job posts such as extraordinaly high remuneration, lower education & work experience requirements, etc. While these general stereotypes allow us to use a common-sense approach to fraud-detection, it is not always foolproof. Clever scammers disguise their posts in such a way so as to fool even the most discerning of candidates. Incidentally, algorithms also struggle to identify fraud in these gray areas.

## **Objectives**

1. Test general stereotypes regarding fake job posts. These stereotypes consist of:
   1. Fake job posts have poor grammar
   2. Fake job posts have higher remuneration
   3. Fake job posts prioritize Cash bonuses
   4. Fake job posts have lower educational & work requirements
   5. Fake job posts have a higher frequency of Clickbait words.
2. Detect and investigate patterns in the data against fraud/non-fraud targets
3. Gain insight into the behavior of scammers & fraudsters
4. Create a framework & methodology for fake job classification. This consists of:
   1. Creating data pre-processing & cleansing techniques
   2. Creating a system for model-driven decision making
   3. Creating a systematic framework for feature engineering
5. Create models to detect fraudulent posts

## **Assumptions regarding Dataset**

1. Typos, spelling errors, and grammatical mistakes reflect ground truth and are not a result of improper data extraction.
2. Data reflects objective truth on job portals, and has not been doctored.
3. NaN columns reflect incomplete or inapplicable post information. Some job posts leave the salary\_range field blank in favour of posting the salary in the description field.
4. All posts in the dataset are authentic, and not engineered/doctored.

## **Project Dependencies**

|  |  |
| --- | --- |
| Name of package/framework/library | Purpose |
| pandas | Data manipulation |
| numpy | Numerical operations |
| tensorflow | Deep Learning |
| spacy | Natural Language Processing |
| regex | Natural Language Processing |
| os | Administrative |
| matplotlib | Graphing & Plotting |
| sklearn | Machine Learning |
| wordcloud | Wordcloud Generation |
| string | Natural Language Processing |
| seaborn | Graphing & Plotting |
| PIL | Image creation |
| nltk | Natural Language Processing |
| dmba | Graphing & Plotting |
| fuzzywuzzy | Natural Language Processing |
| util | Self-created file containing all helper functions |
| keras | Deep Learning |
| dtreeviz | Graphing & Plotting |
| imblearn | Up/Down-sampling |
| lightgbm | Machine Learning |
| tqdm | Natural Language Processing |
| textblob | Sentiment Analysis |
| textstat | Text Statistics |

*Total number of packages: 22*

## **Dataset Description**

The dataset was obtained from the University of California Irvine (UCI) Machine Learning Repository. It consists of 17 variables, including the target variable: fraudulent.

|  |  |  |
| --- | --- | --- |
| Feature Name | Measurement Level | Python Data Type |
| title |  | object |
| location | Nominal | object |
| department | Nominal | object |
| salary\_range | Ordinal | object |
| company | Nominal | object |
| description |  | object |
| requirements |  | object |
| benefits |  | object |
| telecommuting | Binary | object |
| has\_company\_logo | Binary | object |
| has\_questions | Binary | object |
| employment\_type | Nominal | object |
| required\_experience | Ordinal | object |
| required\_education | Ordinal | object |
| industry | Nominal | object |
| function | Nominal | object |
| fraudulent | Binary | object |

Limitations of the Dataset

1. Small size: The dataset only consists of 17.8k records. In problems like this, it is typical to have large datasets consisting of hundreds of thousands of records.
2. Noisy data: The textual features have a lot of typos, unwanted HTML elements, unwanted metadata tags, etc. This makes textual analysis difficult, and requires extensive data cleansing & processing.
3. Imbalanced Dataset: The majority class (non-fraudulent) makes up about 95% of the dataset. The imbalanced nature of the dataset frustrates model-building & evaluation.
4. Empty/NaN Fields: A large number of records have some empty fields. These fields are either left empty intentionally by job posters, or have been left empty due to some mistake. The empty fields introduce yet another level of complexity to the task of detecting fraud, and requires imputation.

# **Data Engineering**

The data engineering undertaken was closely linked to the modelling/analytics approach. It consisted of creating features to determine text/post characteristics, as well as sentiment analysis. Imputation was also carried out in order to make the data compatible with the machine learning algorithms used. Finally, up-sampling was done through the imblearn library using SMOTE (Synthetic Minority Oversampling Technique).

*The complete code-set for feature engineering can be found in the Feature Engineering Jupyter notebook accompanying this documentation.*

Engineered Variables

Major classes of variables:

1. **Text characteristics:** whitespaces, commas used, consecutive punctuation
2. **Sentiment and Text Analysis:** Fleisch reading score, Sentiment Polarity score
3. **Post Completeness:** Missing attributes
4. **Post Characteristics:** employment type, industry, function

## **Text Cleaning & Processing (Lemmatization)**

The text features had to be processed in order to remove unwanted elements. This was done using the spacy, nltk and regex libraries. Punctuation marks, unwanted metadata elements, and HTML tags were filtered out. The cleaned text features were appended back to the dataset with the suffix \_cleaned.

Lemmatization was also carried out in order to simplify the text. **Lemmatization is a process of simplifying multiple inflections of a word into a single unit.** For example, the words – *plays*, and *playing* will be lemmatized into the simpler *play*. It is important to note that this might introduce bias into the data – as cultural differences in the form of words used will be glossed over through lemmatization.

*Feature Engineering #4: Text Field Clean up*

# Adding HTML elements to stopwords

stop\_words.extend(['etc', 'u', 'ul', 'br', 'b', 'i', 'false', 'li', 'w', 'ww', 'www', 'p', 'LsdException', 'nofollow', 'amp', 'grid', 'accent'])

clean\_up\_cols = ['company\_profile', 'requirements', 'description', 'benefits']

for col in clean\_up\_cols:

    # name cleaned up column

    cleaned\_col\_name = col + '\_cleaned'

    # initiate empty column

    data[cleaned\_col\_name] = ''

    for i in range(len(data[col])):

        # if data is na, place empty string

        if pd.isna(data[col][i]):

            data[cleaned\_col\_name][i] = np.NAN

        else:

            # dont need entire nlp pipeline so parser and ner are disabled

            text\_doc = nlp(data[col][i], *disable*=['parser', 'ner'])

            # text is lemmatized, converted to lowercase, and separated by whitespace

            text = [tok.lemma\_.lower().strip() for tok in text\_doc]

            # stopwords and punctuation marks are filtered out

            text = [tok for tok in text if tok in words and tok not in stop\_words and tok not in punctuations and tok.isnumeric() is False]

            # the resulting token list is joined back into a sentence

            text =  ' '.join(text)

            data[cleaned\_col\_name][i] = text

## **Text/Post Characteristics Variables**

Missing Indicators

Missing indicators were created for the major text features before imputation was carried out. These indicators were encoded as binary integer values (1/0).

*Missing indicator feature creation*

# Creating missing indicators for post attributes

attributes = ['industry', 'function', 'requirements', 'benefits', 'description' ,'employment\_type', 'salary\_range', 'location']

for col in attributes:

    new\_col =  'has\_' + col

    data[new\_col] = data[col].isna().astype(int)

Imputation

Imputing missing values in this dataset is challenging due to the fact that it is hard to infer the value of a missing field from text fields. Furthermore, the dataset consists of categorical/nominal level variables entirely – further frustrating the process of imputation as simple regression imputation cannot be used. In light of this fact, the Iterative Imputer from the sci-kit learn package was used.

The Iterative Imputer models each feature with a missing value as a function of the other features. It does in a round-robin fashion until the maximum iteration as specified by the user.

*Imputation of missing features*

imputer = IterativeImputer(*random\_state* = 11, *initial\_strategy* = 'most\_frequent', *imputation\_order* = 'descending')

X = imputer.fit\_transform(X)

It must be noted that only the text/post characteristics features were imputed. Text itself was never imputed – this is a particularly difficult task, and without a Transformer model like GPT-3, it is impossible.

Additionally, before imputation: any feature with more than 60% missing values were dropped.

Sentiment Analysis

I wanted to investigate any potential differences in tone between fraudulent & non-fraudulent job posts. In order to do this, I computed the polarity score using the TextBlob package.

*Sentiment Score Feature Engineering*

clean\_up\_cols = ['company\_profile\_cleaned', 'requirements\_cleaned', 'description\_cleaned', 'benefits\_cleaned']

for col in clean\_up\_cols:

    # name cleaned up column

    col\_name = col + '\_sentiment'

    # initiate empty column

    data[col\_name] = 0.0

    for i in range(len(data[col])):

        # if data is na, place NAN object

        if pd.isna(data[col][i]):

            data[col\_name][i] = np.NAN

        else:

            # assign read score to field

            data[col\_name][i] = TextBlob(data[col][i]).sentiment.polarity

Reading Score & Average number of words per sentence

I also wanted to test differences in readability between fraudulent & non-fraudulent posts. In order to do this, I computed the Flesch Reading Ease Score through the textstat library.

The Flesch reading score measures the readability of a given text on a score of 0-100. While the maximum value is 121.22, there is no limit on how low the score can be. Negative values are valid.

Flesch Reading Scores

|  |  |
| --- | --- |
| Score | Readability |
| 0-29 | Very Confusing |
| 30-49 | Difficult |
| 50-59 | Fairly Difficult |
| 60-69 | Standard |
| 70-79 | Fairly Easy |
| 80-89 | Easy |
| 90-100 | Very Easy |

I also computed the average number of words per sentence in the text features through the textstat library. This was done in order to test whether fraudulent posts had more run-on sentences.

clean\_up\_cols = ['company\_profile\_cleaned', 'requirements\_cleaned', 'description\_cleaned', 'benefits\_cleaned']

for col in clean\_up\_cols:

    # name cleaned up column

    col\_name = col + '\_words\_per\_sentence'

    # initiate empty column

    data[col\_name] = 0.0

    for i in range(len(data[col])):

        # if data is na, place NAN object

        if pd.isna(data[col][i]):

            data[col\_name][i] = np.NAN

        else:

            # assign read score to field

            data[col\_name][i] = textstat.words\_per\_sentence(data[col][i])

Collapsing/merging categories

The industry feature had many different industries – which if one-hot encoded would result in a sparse matrix with many categorical columns. In order to combat this, similar industries were grouped together. Industries which made less than 1% of the total count of industries were grouped under a “Miscellaneous” category.

# Collapsing together Health industries

data['industry'][data['industry'][data['industry'].str.contains('Health', *na* = False)].index] = 'Hospital & Healthcare'

# Collapsing together Banking & Financial industries

data['industry'][data['industry'][data['industry'].str.contains('Banking', *na* = False)].index] = 'Banking and Financial Services'

data['industry'][data['industry'][data['industry'].str.contains('Financial', *na* = False)].index] = 'Banking and Financial Services'

# Collapsing together 'Computer' industries

data['industry'][data['industry'][data['industry'].str.contains('Computer', *na* = False)].index] = 'Computer/IT/Software'

# Collapsing together IT & Information industries industries

data['industry'][data['industry'][data['industry'].str.contains('Information', *na* = False)].index] = 'Computer/IT/Software'

data['industry'][data['industry'][data['industry'].str.contains('Internet', *na* = False)].index] = 'Computer/IT/Software'

# Collapsing together Animation industry with Motion Pictures

data['industry'][data['industry'][data['industry'].str.contains('Animation', *na* = False)].index] = 'Motion Pictures & Film'

# Collapsing together Trade industry with EXIM

data['industry'][data['industry'][data['industry'].str.contains('Trade', *na* = False)].index] = 'Import and Export'

# Collapsing all management professions/industries into one Management category

data['industry'][data['industry'][data['industry'].str.contains('Management', *na* = False)].index] = 'Management'

# Collapsing all Education industries into one category

data['industry'][data['industry'][data['industry'].str.contains('Education', *na* = False)].index] = 'Education'

# Collapsing industries with count less than 100 into Miscellaneous category

for industry in data['industry'].value\_counts()[data['industry'].value\_counts() <= 100].index.tolist():

    data['industry'][data['industry'][data['industry'].str.contains(industry, *na* = False)].index] = 'Miscellaneous'

## **Term-Document Matrix & TF-IDF**

A term-document matrix (also known as a Document-Term matrix) describes the frequency of terms/words occurring in a document/text. All the cleaned textual features were combined together and passed to the Count Vectorizer function from sklearn in order to create a Term-Document matrix.

Infrequent terms with a proportion of 0.1 were discarded so as to reduce the size of the resulting Term-Document matrix. Without filtering for infrequent terms, the size of the matrix would be **17,880 X 1,114,558**.

After the creation of the Term-Document matrix, the matrix was passed to the Tfidf transformer from sklearn in order to compute the Term Frequency – Inverse Document Frequency (TF-IDF) scores. The TF-IDF is a statistic that measures how important a word is to a given corpus/document/text. The higher the score, the more important the word.

## **Upsampling with SMOTE**

In light of the imbalanced nature of the dataset, up-sampling was done through the imblearn library using SMOTE (Synthetic Minority Oversampling Technique). SMOTE is a technique for generating synthetic samples for the minority class in a dataset. It can be used to balance & augment an imbalance dataset, allowing for better modelling results. However, using SMOTE introduces bias & error into the final conclusion.

## **One-hot encoding**

Lastly, dummy variables were created for each categorical feature.

X = pd.get\_dummies(data\_trad[[col for col in data\_trad if col not in (

## **Clickbait words & Clickbait Ratio**

Clickbait words are words designed to attract readers to a post. They are titled vaguely, or in outrageous terms, and are often misleading. Scammers use clickbait terms to draw potential victims in, and frequently use these words in their posts.

I have used my own interpretation of clickbait words based on the data I sampled.

clickbait\_words = ['earn', 'home', 'week', 'remote', 'daily', 'cash', 'high school', 'no degree']

I calculated the count of clickbait words in each text field. This was then used to compute the Clickbait Ratio: the count of clickbait words in a field to the total length of that field.

## **Datasets Created**

Four major datasets were created:

1. Dataset with only Text/Post Characteristics
2. Term-Document Matrix (Counts): This dataset was only created a by-product of the text transformation process. It was briefly used with some models to determine utility, but ultimately deemed non-useful compared to the TF-IDF matrix.
3. TF-IDF Matrix
4. TF-IDF Matrix with Text/Post Characteristics

Two versions of each dataset were created: with up-sampling, and with imbalance preserved.

## **Data Summary**

Summary Statistics for Feature Set



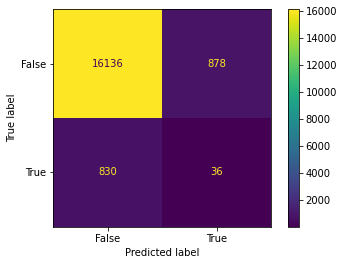
Please note: The entire summary table cannot be displayed here as it consists of 45 tables. The summary statistics can be accessed in the Complete EDA Jupyter Notebook file accompanying this documentation, as well as the data\_summary csv file.

# **Modelling**

## **Baseline Results**

All model results will be compared to baseline results which consists of naively guessing the majority class according to the proportion of the class. For instance, suppose a dataset consists of the objects (A, A, B). A naïve method of prediction would be to guess A 2/3rd of the time. This would yield an accuracy of 2/3 overall, and although the strategy is not sophisticated – it yields a baseline level of accuracy which models would need to beat. Failure to do so means no utility can be gained from the model apart from explanatory modelling.

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 90.447% |
| True Positive Rate (Sensitivity) | 4.157% |
| True Negative Rate (Selectivity) | 94.8395% |
| False Positive Rate | 5.1605% |
| False Negative Rate | 95.8430% |
| Precision | 3.9387% |
| F1-Score | 4.0449% |

  
Confusion Matrix of Naïve Guessing

## **Modelling Approach**

An iterative approach was taken to modelling & analytics. The main steps consisted of:

1. Exploring the dataset to find patterns
2. Creating models to determine indicators of fraud
3. Using model findings to generate recommendations
4. Revisiting exploration with new findings (restarting process)

In analysing the dataset, I used several approaches. These approaches consisted of:

1. **Bag of words approach:** Word based analysis, fraudulent or not based on word frequency & corresponding Term Frequency – Inverse Document Frequency (TF-IDF) scores.

Fraudulent postings are deliberately structured in such a way as to target susceptible and vulnerable people. This can be used to detect these posts.

1. **Tone:** Sentiment analysis – are fraudulent posts more positive?
2. **Post Characteristics and Completeness:** Are fraudulent posts incomplete? Do they have more whitespaces or grammatical errors?
3. I also manually examined a thousand of the job posts in order to determine the best way to process, clean, and analyse them.

Model Inventory

Four major classes of models were created. All models were trained on both, the up-sampled dataset, and imbalanced dataset. Except for the Variational Autoencoder, all models were only trained on the Text/Post Characteristics dataset.

In total, 15 models were created.

|  |  |
| --- | --- |
| Model Category | Model |
| Naïve Bayes Classifiers | CategoricalNB |
|  | ComplementNB |
| Tree Models | Decision Trees |
|  | Random Forest |
| Gradient Boosted Trees | Histogram Gradient Boosting Classifier |
|  | Light Gradient Boosting Machine Classifier |
| Variational Autoencoder |  |

Evaluation Statistic

In light of the imbalanced nature of the dataset, the F1-score was used to evaluate the models, along with Accuracy, Precision & Recall.

Accuracy would fail to depict true model performance; therefore, a balanced measure (the F1-Score) was chosen. I contemplated using the Brier Score, but ultimately decided to choose the simpler F1-Score.

Permutation Feature Importance was used to interpret how models were weighing features. This is a model agnostic tool of determining the most important features for a given model. Even if a feature is deemed important by one model, it is not necessary to be deemed important by another model.

## **Naïve Bayes Classifier**

Naïve Bayes Classifiers are probability-based classifiers based on Bayes Theorem.

Model Assumptions:

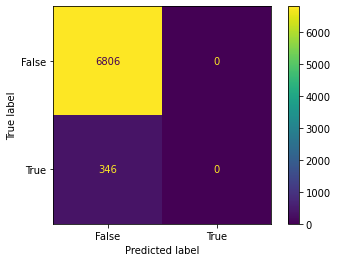
1. Dataset consists of records which are independent of each other.
2. Each feature makes an equal contribution to the outcome.

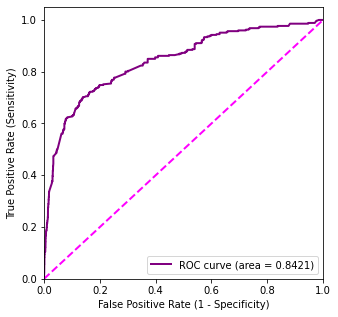
Categorical Naïve Bayes

The Categorical Naïve Bayes is sci-kit learn’s vanilla implementation of a Naïve Bayes Classifier. The class\_prior was used in order to adjust the model’s predictions for the class imbalance. Furthermore, the min\_categories column was used for features which didn’t have a category in the training dataset. This ensured that no error would be thrown by the model.

**Performance on imbalanced dataset**

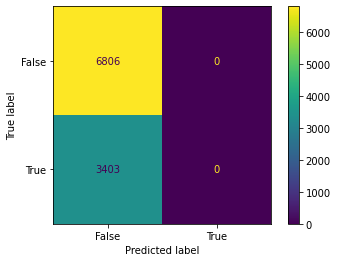
|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 95.1622% |
| True Positive Rate (Sensitivity) | 0% |
| True Negative Rate (Selectivity) | 100% |
| False Positive Rate | 0% |
| False Negative Rate | 100% |
| Precision | N/A |
| F1-Score | N/A |

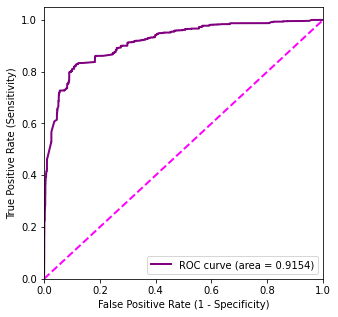
**  
Confusion Matrix on imbalanced dataset**

**  
ROC Curve on imbalanced dataset**It can be seen that the AUC is a bad evaluation metric for imbalanced datasets.  
Even with such poor classification, the AUC of the classifier is as high as 84%.

**Performance on up-sampled dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 66.6667% |
| True Positive Rate (Sensitivity) | 4.157% |
| True Negative Rate (Selectivity) | 94.8395% |
| False Positive Rate | 5.1605% |
| False Negative Rate | 95.8430% |
| Precision | 3.9387% |
| F1-Score | 4.0449% |

**  
Confusion Matrix on up-sampled dataset**

****

**ROC Curve on up-sampled dataset**

**Comments**

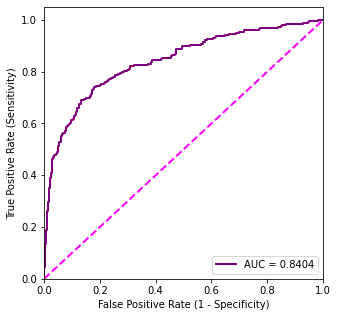
The Categorical Naïve Bayes Classifier performed very poorly on both versions of the text characteristics dataset. Due to the poor performance of the model, feature importance was not computed, and interpretation of the model was skipped.

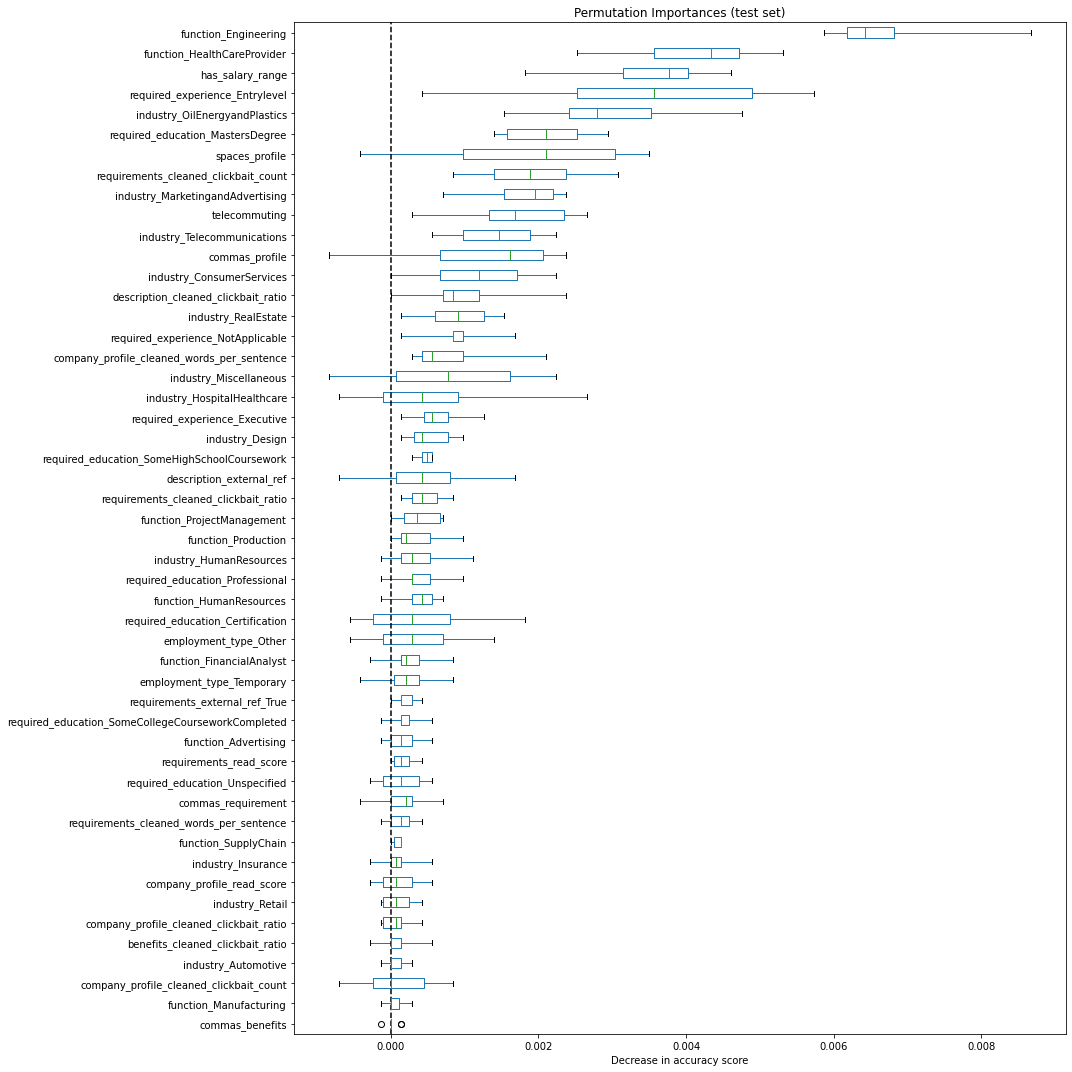
Complement Naïve Bayes

The Complement Naïve Bayes Classifier was developed in order to work with imbalanced datasets – it was designed to overcome the severe assumptions made by a normal Multinomial Naïve Bayes Classifier. The name Complement is derived from the fact that instead of calculating the probability of an item belonging to a class, the probability of an item **not** belonging to a class is calculated.

**Performance on imbalanced dataset**

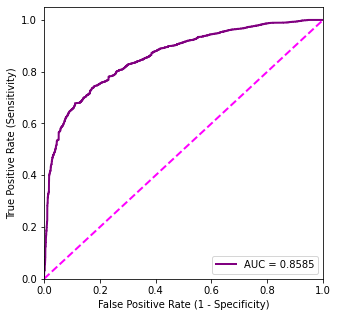
|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 72.7209% |
| True Positive Rate (Sensitivity) | 79.4798% |
| True Negative Rate (Selectivity) | 72.3773% |
| False Positive Rate | 27.6227% |
| False Negative Rate | 20.5202% |
| Precision | 12.7610% |
| F1-Score | 21.9912% |

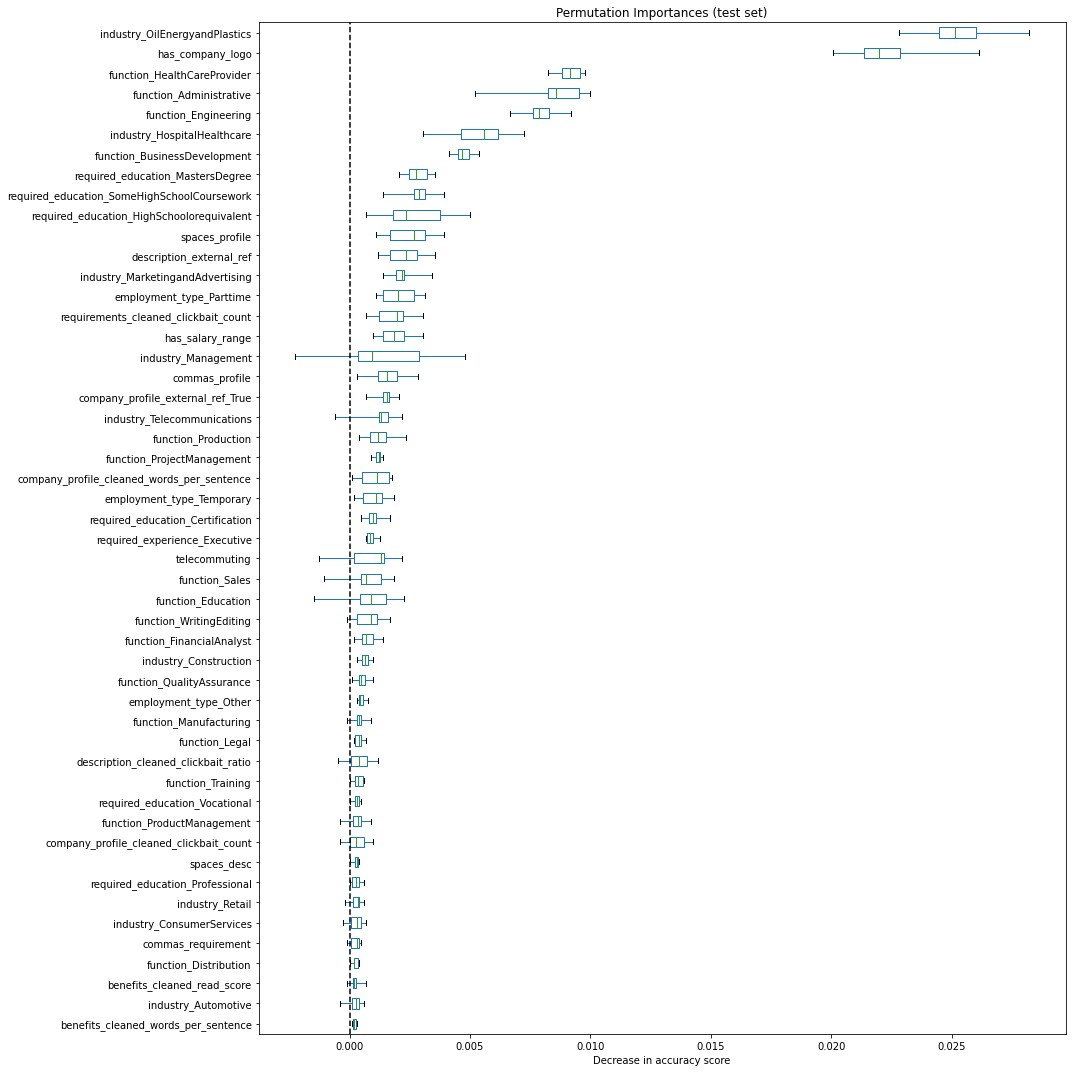
**  
ROC Curve on imbalanced dataset**

**  
Permutation Feature Importance on imbalanced dataset**

**Performance on up-sampled dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 74.6302% |
| True Positive Rate (Sensitivity) | 81.5163% |
| True Negative Rate (Selectivity) | 71.1872% |
| False Positive Rate | 28.8128% |
| False Negative Rate | 18.4837% |
| Precision | 58.5850% |
| F1-Score | 68.1740% |

**  
ROC Curve on up-sampled dataset**

**  
Permutation Feature Importance on up-sampled dataset**

**Comments**

Although the Complement Naïve Bayes classifier performed much better than the vanilla implementation, it still had a fairly abysmal F1-score. It struggled to distinguish between fraudulent & non-fraudulent job posts with a fairly high false-positive rate of 28%.

The model mainly relied on the Oil industry, and Health-care provider function variables to determine if a post was fraudulent. It also determined that a post with company logo was more likely to be a fraud.

## **Tree Models**

Tree models are highly versatile models capable of dealing with outliers without treatment. Some tree models can also handle NaN values natively, by assigning them to branches without imputation.

I used 5-fold Random Search Cross validation with 60 iterations to search through a grid of parameters, and determine the most optimal settings.

Decision Trees

parameters\_RF = {

    'min\_impurity\_decrease': np.arange(0, 1, 0.1),

    #'ccp\_alpha': np.arange(0, 0.6, 0.1),

    'criterion': ['log\_loss', 'entropy'],

    'max\_depth': range(2, 10, 3),

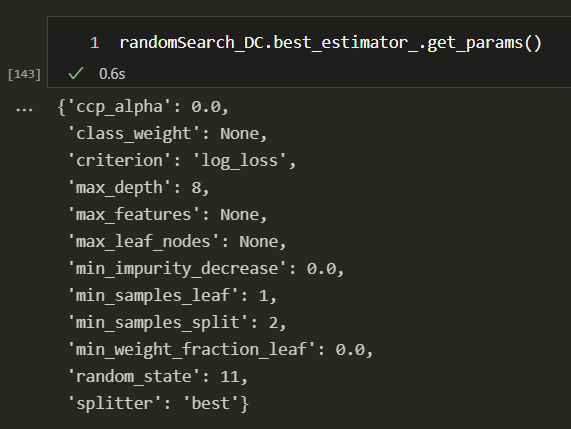
    #'min\_samples\_leaf': np.arange(0.01, 0.9, 0.01),

    'min\_samples\_split': range(2, 50, 2),

    'max\_features': [None, 'sqrt'],

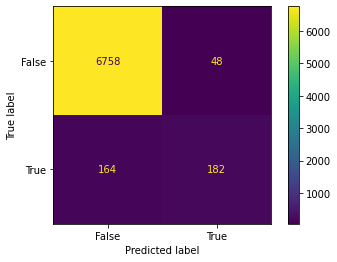
    'random\_state': [11]

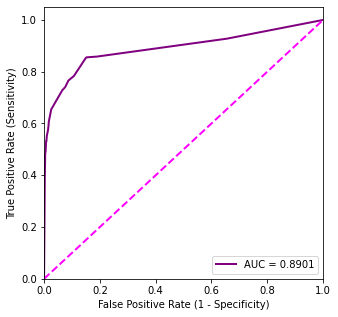
}

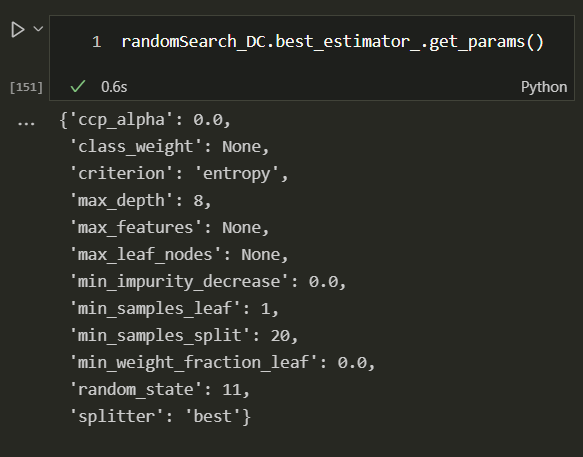
**  
Best Parameters for imbalanced dataset**

**Performance on imbalanced dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 97.0358% |
| True Positive Rate (Sensitivity) | 52.6012% |
| True Negative Rate (Selectivity) | 99.2947% |
| False Positive Rate | 0.7053% |
| False Negative Rate | 47.3988% |
| Precision | 79.1304% |
| F1-Score | 63.1944% |

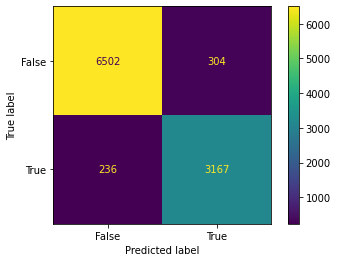
**  
Confusion Matrix on imbalanced dataset**

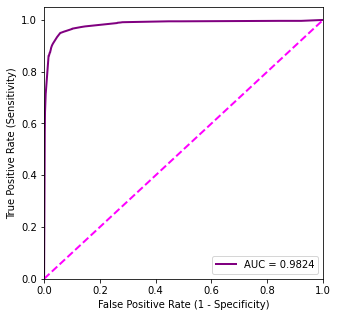
**  
ROC Curve on imbalanced dataset**

**  
Best Parameters for up-sampled dataset**

**Performance on up-sampled dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 94.7105% |
| True Positive Rate (Sensitivity) | 93.0649% |
| True Negative Rate (Selectivity) | 95.5334% |
| False Positive Rate | 4.4666% |
| False Negative Rate | 6.9351% |
| Precision | 91.2417% |
| F1-Score | 92.1443% |

**  
Confusion Matrix on up-sampled dataset**

**  
ROC Curve on up-sampled dataset**

**Comments**

The Decision tree performed very well on the up-sampled dataset. It was able to distinguish between fraudulent & non-fraudulent posts in a healthy manner. However, it was noticed that while running the Decision tree – it was highly prone to the training sample. In other words, the trees created on different runs differed a lot due to high variance. Furthermore, the tree was only able to perform well due to the huge depth to which it was created. Due to these reasons, it was determined that a Decision tree would not be stable enough for fraud detection.

Very egregious overfitting is most likely occurring, and the tree is likely to fail.

Random Forests

Random Forests comprise of several trees, and the prediction is made through a voting mechanism whereby the prediction with the most votes (from trees) wins.

parameters\_RF = {

    'n\_estimators': range(50, 200, 20),

    #'min\_impurity\_decrease': np.arange(0, 1, 0.1),

    'criterion': ['log\_loss', 'entropy'],

    'max\_depth': range(2, 30, 5),

    #ccp\_alpha': np.arange(0, 0.6, 0.1),

    #'min\_samples\_leaf': np.arange(0.1, 0.6, 0.1),

    'min\_samples\_split': range(2, 20, 2),

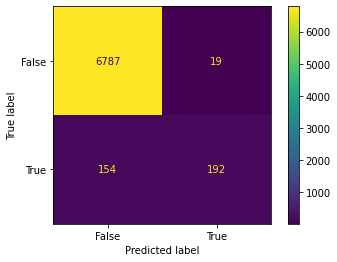
    'max\_features': [None, 'sqrt', 'log2'],

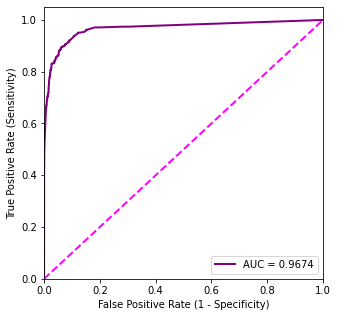
    'random\_state': [11]

}

**Performance on imbalanced dataset**

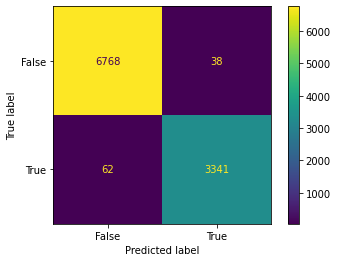
|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 97.5811% |
| True Positive Rate (Sensitivity) | 55.4913% |
| True Negative Rate (Selectivity) | 99.7208% |
| False Positive Rate | 0.2792% |
| False Negative Rate | 44.5087% |
| Precision | 90.9953% |
| F1-Score | 68.9408% |

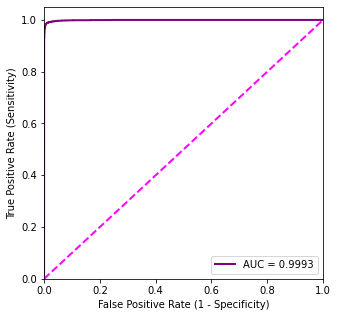
**  
Confusion Matrix on imbalanced dataset**

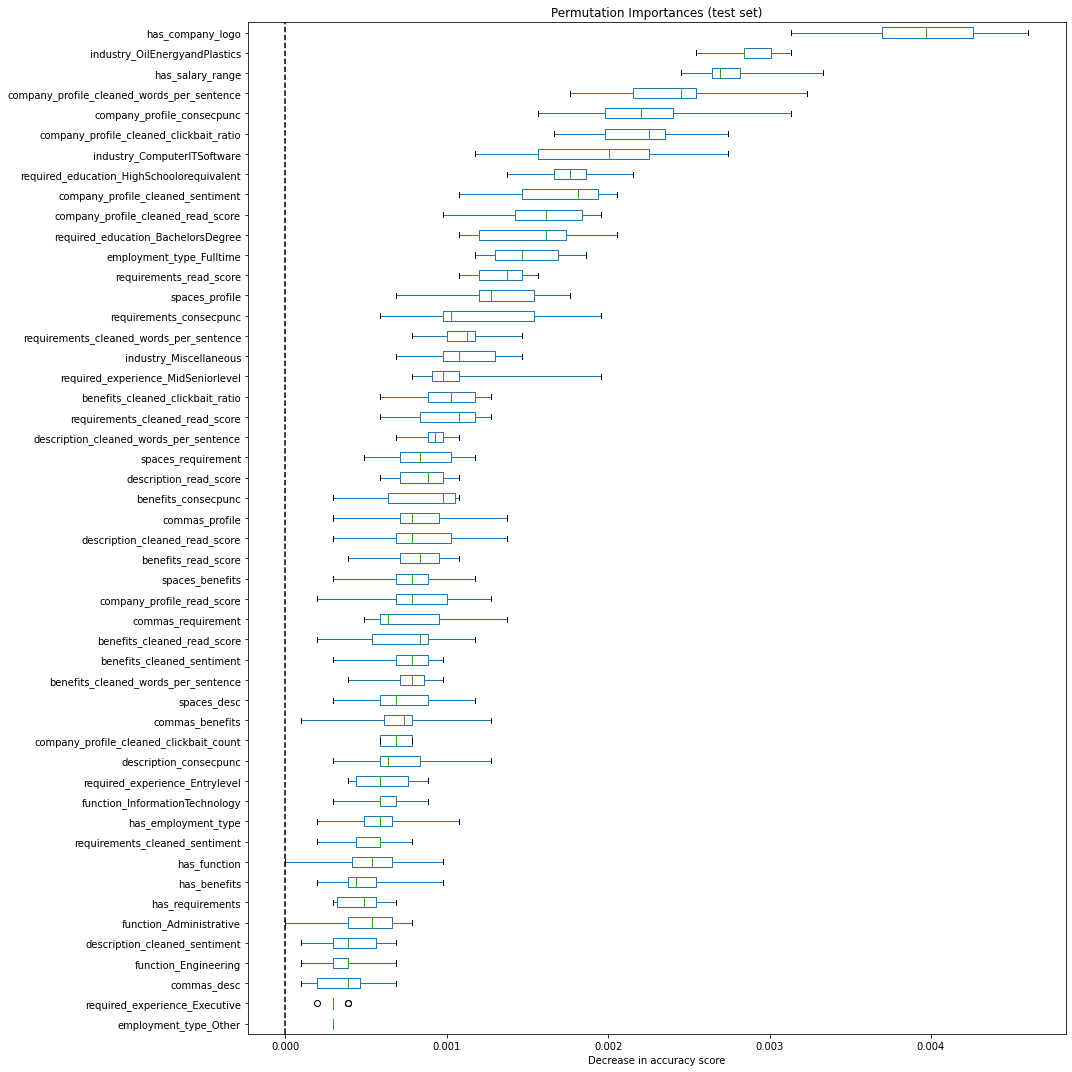
  
**ROC Curve on imbalanced dataset**

**Performance on up-sampled dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 99.0205% |
| True Positive Rate (Sensitivity) | 98.1781% |
| True Negative Rate (Selectivity) | 99.4417% |
| False Positive Rate | 0.5583% |
| False Negative Rate | 1.8219% |
| Precision | 98.8754% |
| F1-Score | 98.5255% |

  
**Confusion Matrix on up-sampled dataset**

**  
ROC Curve on up-sampled dataset**

**  
Permutation Feature Importance on up-sampled dataset**

**Comments**

The Random Forest performed very admirably on the up-sampled dataset achieving the highest F1-Score of all tree models. It considered the sentiment score, reading score, and high-school educational requirement features as important. It also prioritized the has\_company\_logo, and has\_salary\_range missing indicators as important features. Posts which have both these features have a higher proportion of fraudulent targets.

## **Gradient Boosted Trees**

Gradient Boosting is a greedy machine learning algorithm used to minimize the differences between predicted & actual values. On each iteration stage, a tree is built to bridge the gap between the difference in prediction & reality. As a result, overfitting can occur very easily and a proper regularization parameter is required. These trees can work well with imbalanced & complex datasets (Naude, Adebayo and Nanda, 2022).

Histogram Gradient Boosted Tree

Histogram Gradient Boosted Trees (HGBT) are sci-kit learn’s implementation of gradient boosted trees inspired by LGBM. These models are capable of handling NaN values natively, and are also well-suited to work with imbalanced datasets. However, in order to avoid ambiguity – the NaN values were imputed with the Iterative Imputer – even though the model can handle them natively.

parameters = {

 'max\_iter': [1000, 1100, 1200, 1250, 1300],

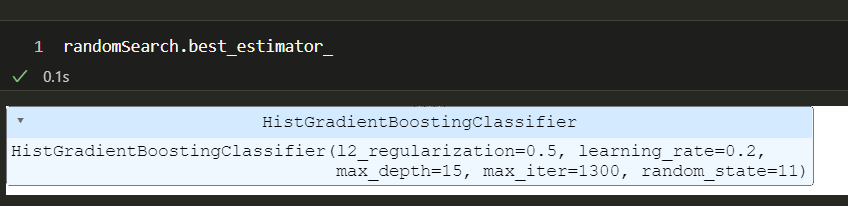
'learning\_rate': [0.1, 0.15, 0.2, 0.25],

 'max\_depth' : [10, 15, 20, 25, 30],

 'l2\_regularization': [0.25, 0.5, 0.75, 1, 1.5],

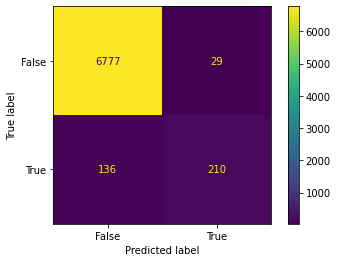
 'random\_state' : [11],

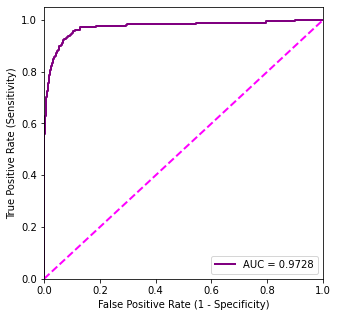
 }

  
**Best parameters for imbalanced dataset**

**Performance on imbalanced dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 97.6930% |
| True Positive Rate (Sensitivity) | 60.6936% |
| True Negative Rate (Selectivity) | 99.5739% |
| False Positive Rate | 0.4261% |
| False Negative Rate | 39.3064% |
| Precision | 87.8661% |
| F1-Score | 71.7949% |

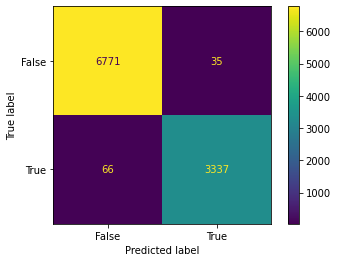
**  
Confusion Matrix on imbalanced dataset**

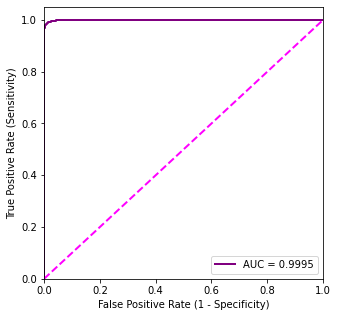
**  
ROC Curve on Imbalanced Dataset**

**  
Permutation Feature Importance on imbalanced dataset**

**Performance on up-sampled dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 99.0107% |
| True Positive Rate (Sensitivity) | 98.0605% |
| True Negative Rate (Selectivity) | 99.4857% |
| False Positive Rate | 0.5143% |
| False Negative Rate | 1.9395% |
| Precision | 98.9620% |
| F1-Score | 98.5092% |

**  
Confusion Matrix on up-sampled dataset**

**  
ROC Curve on up-sampled dataset**

**  
Permutation Feature Importance on up-sampled dataset**

**Comments**

The Histogram Gradient Boosted Tree performed very well on the dataset as expected. Performance was modest even without up-sampling, and the model was achieving good discrimination between fraudulent & non-fraudulent posts. The model prioritized the average number of words per sentence & clickbait ratio as important features. It also considered readability and sentiment score important features.

Light Gradient Boosting Machine

Light Gradient Boosting Machine (LGBM) is an industry-standard tree model used for machine learning tasks. It can deal with NaN values innately, are light-weight, fast and portable.

fit\_params = {

            "eval\_metric" : "logloss",

            "eval\_set" : [(valid\_X, valid\_Y), (train\_X, train\_Y)],

            'eval\_names': ['valid','train'],

            'feature\_name': 'auto',

            #'feval' :

            #'categorical\_feature': X.select\_dtypes("int8").columns.tolist()

           }

parameters\_lgbm = {

 'max\_depth': [-1],

 'num\_leaves': [5, 10, 15, 20, 25, 30, 35],

 'n\_estimators' : [500, 1000, 1500, 2000],

 'lambda\_l1': [0.75, 1, 1.5],

 'lambda\_l2': [0.75, 1, 1.5],

 'learning\_rate' : [0.015, 0.03, 0.05, 0.1],

 'metric': ['logloss'],

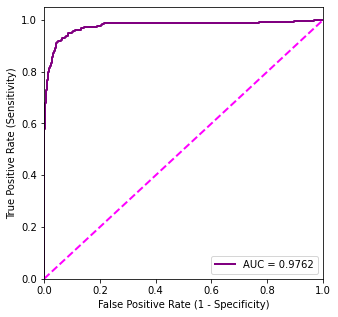
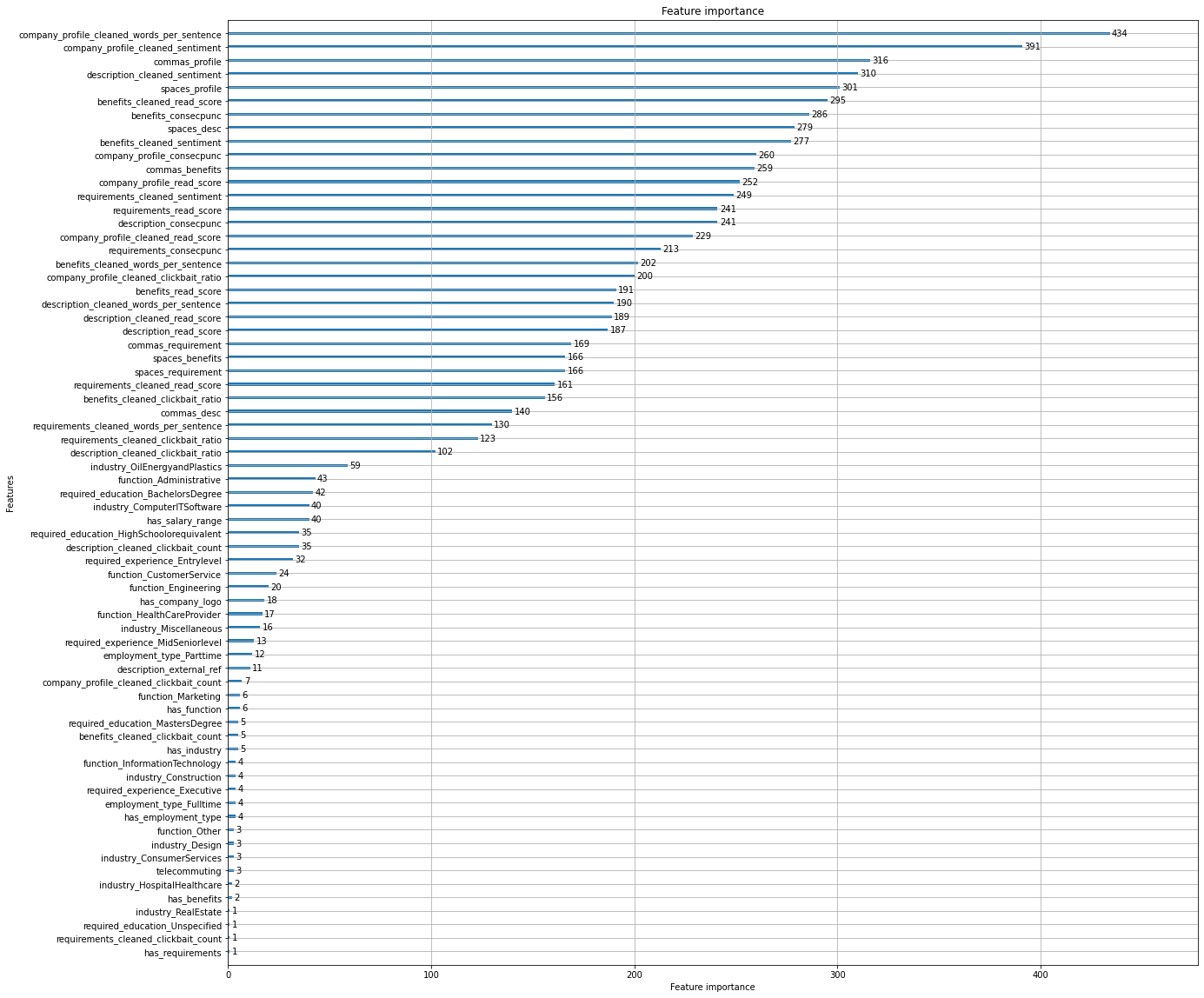
 'colsample\_bytree': [0.3, 0.4, 0.6, 0.7, 0.9],

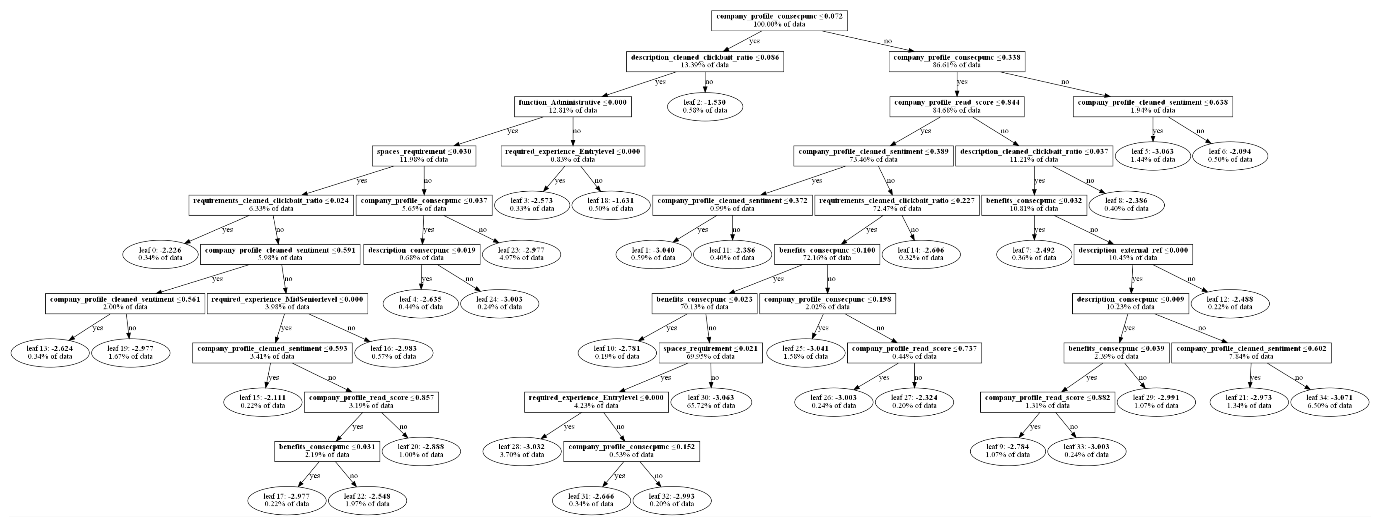
 'random\_state': [11]

 }

**Performance on imbalanced dataset**

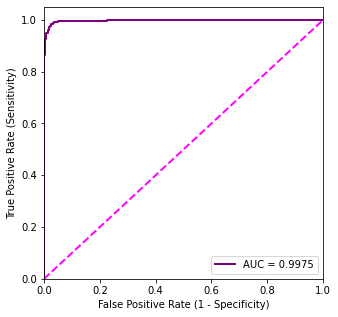
|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 97.9306% |
| True Positive Rate (Sensitivity) | 62.7168% |
| True Negative Rate (Selectivity) | 99.7208% |
| False Positive Rate | 0.2792% |
| False Negative Rate | 37.2832% |
| Precision | 91.9492% |
| F1-Score | 74.5704% |

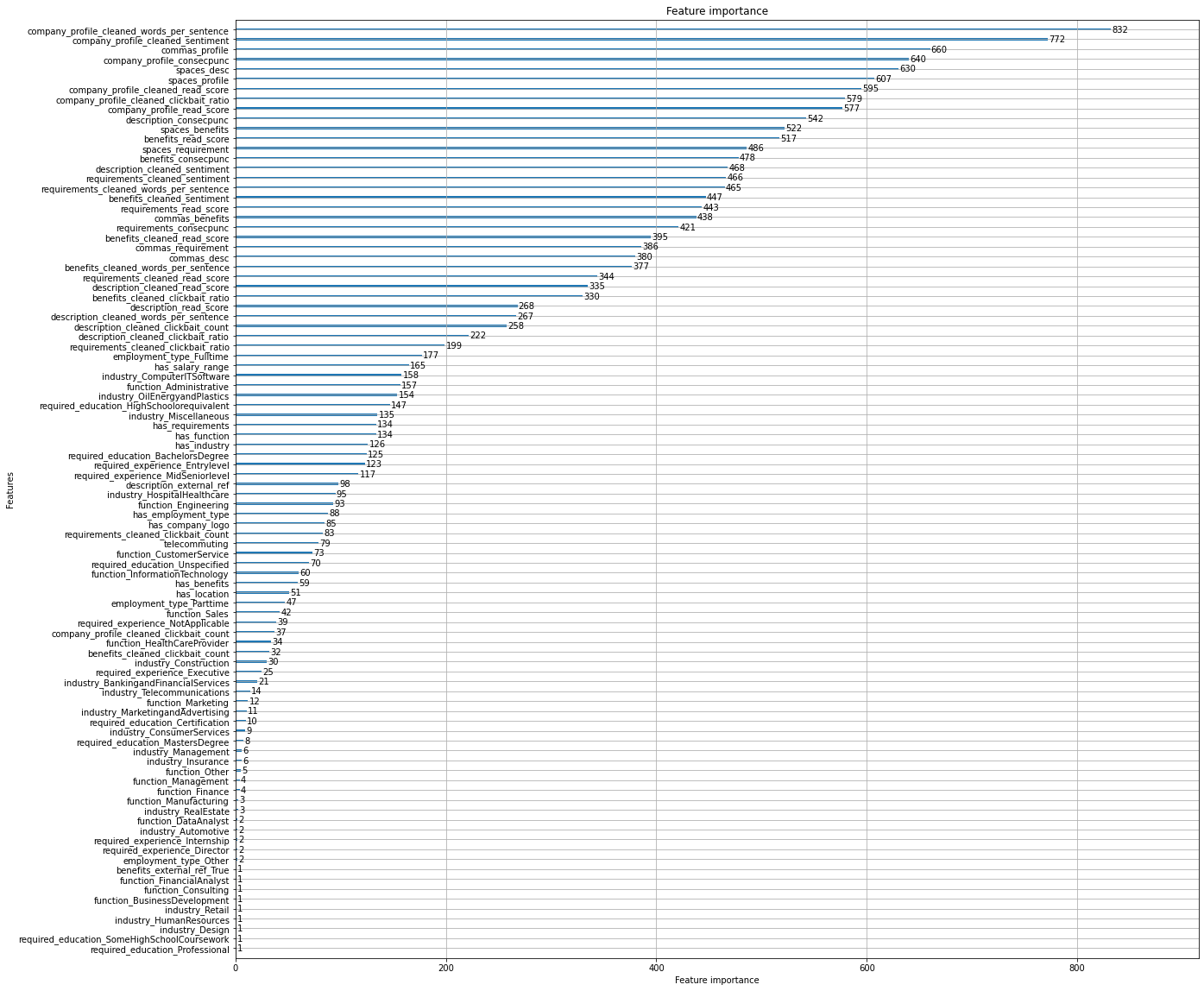
  
**ROC Curve on imbalanced dataset**  
**Feature Importance on imbalanced dataset**

  
**Tree for imbalanced dataset**

**Performance on up-sampled dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 99.0499% |
| True Positive Rate (Sensitivity) | 98.3250% |
| True Negative Rate (Selectivity) | 99.4123% |
| False Positive Rate | 0.5877% |
| False Negative Rate | 1.6750% |
| Precision | 98.8187% |
| F1-Score | 98.5712% |

  
**ROC Curve on up-sampled dataset**

**  
Feature Importance on up-sampled dataset**

**Comments**

The LGBM model performed stunningly well on both datasets – with and without up-sampling. It considered the text features related to grammar, punctuation, and sentiment most important. It also considered clickbait ratio to be an important variable.

## **Neural networks**

Only one class of Neural Networks were tried: the Variational Autoencoder. Later in the project, it was realized that a Generative model would be better suited to the problem at hand.

Variational Autoencoder

A Variational Autoencoder consists of two parts: the encoder & the decoder.

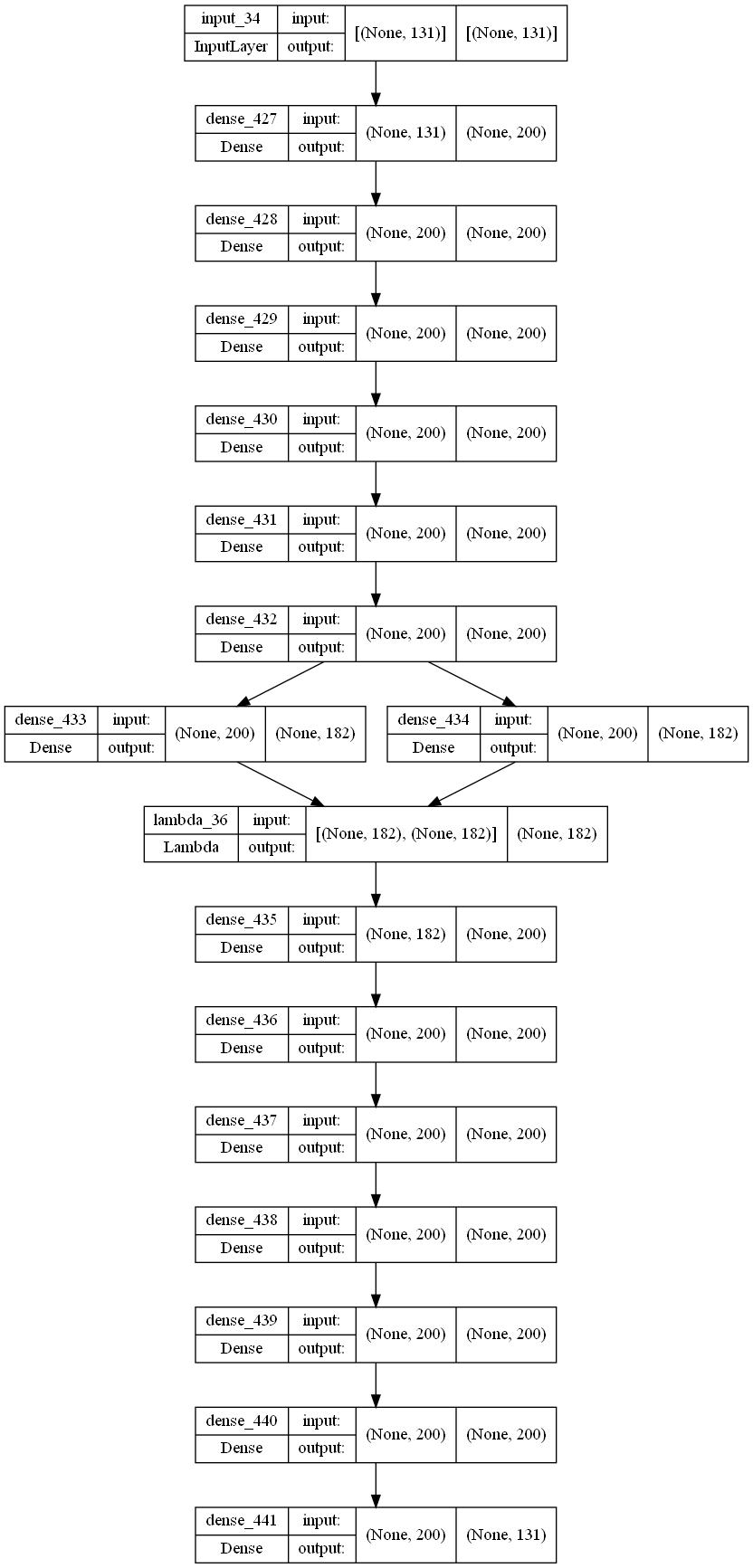
The encoder is responsible for learning the latent space distribution of input features, and the decoder is responsible for recreating them. The inputs pass through a sampling layer, which is used by the decoder. As inputs move through the network, they get compressed – to be then “de-compressed” by the decoder.

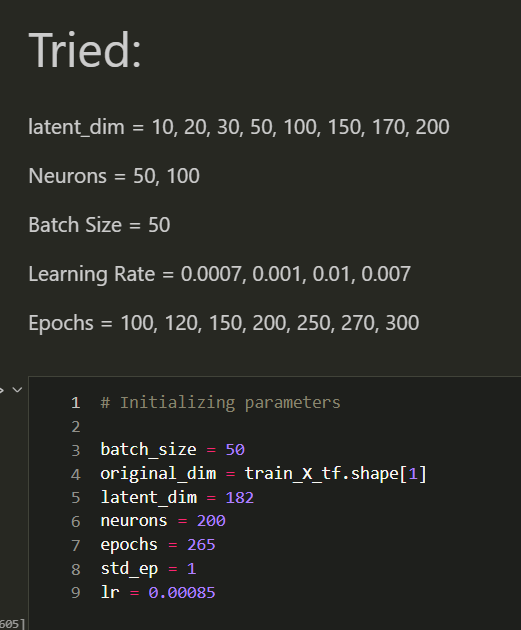
The main hypothesis behind using the VAE was that the distribution for features corresponding to legitimate job posts would be different than the distribution for features corresponding to fraudulent job posts.

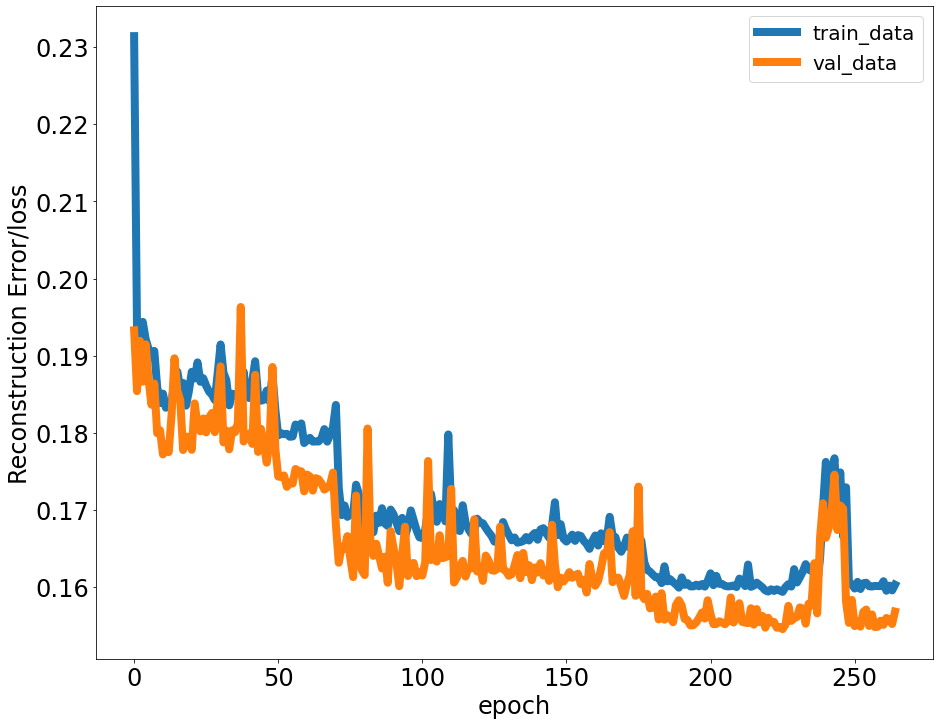
Therefore, the aim was to reconstruct features. Thus, the input & target/label data were both comprised of the feature dataset (TF-IDF matrix). The VAE was trained on all datasets, but it performed best on the TF-IDF matrix.

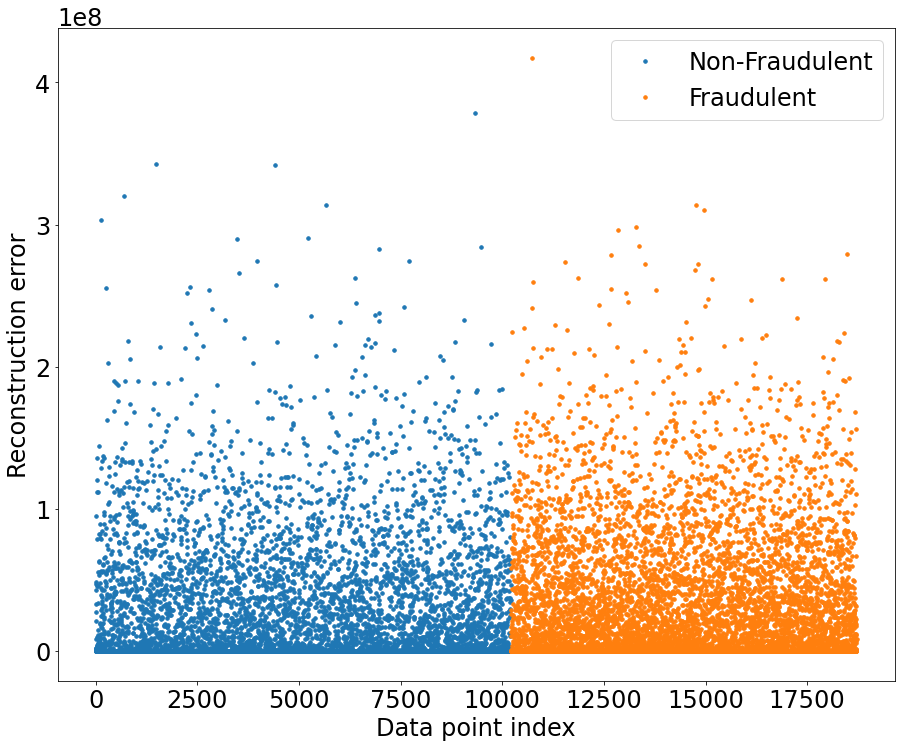
The architecture of the VAE was derived through trial & error. There are 6 hidden layers in both the encoder & decoder, giving a total of 12 hidden layers. In between the encoder & decoder is a sampling layer.

Once the model was trained, the optimal classification threshold was obtained through Youden’s J-statistic.

  
**VAE architecture derived through trial & error**

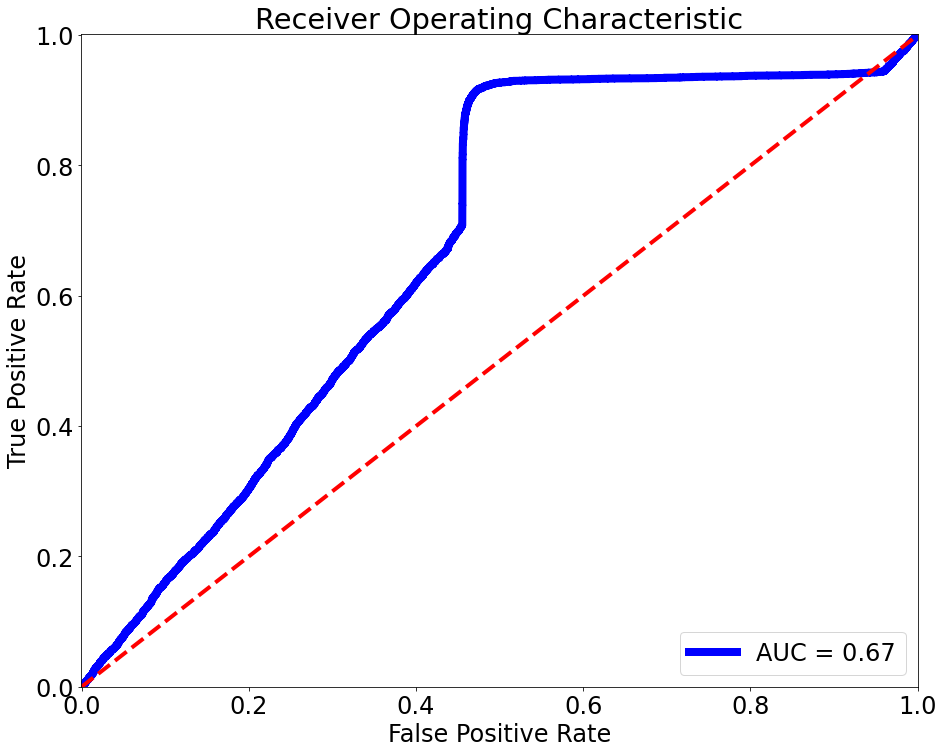
**  
VAE initialization parameters derived through trial & error**

  
**VAE Learning Curves**

**  
Reconstruction error vs target**

**Performance on TF-IDF dataset**

|  |  |
| --- | --- |
| Evaluation metric | Score |
| Accuracy | 70.3607% |
| True Positive Rate (Sensitivity) | 91.6892% |
| True Negative Rate (Selectivity) | 52.5862% |
| False Positive Rate | 47.4138% |
| False Negative Rate | 8.3108% |
| Precision | 61.7089% |
| F1-Score | 73.7693% |

**  
VAE ROC Curve**

**Comments**

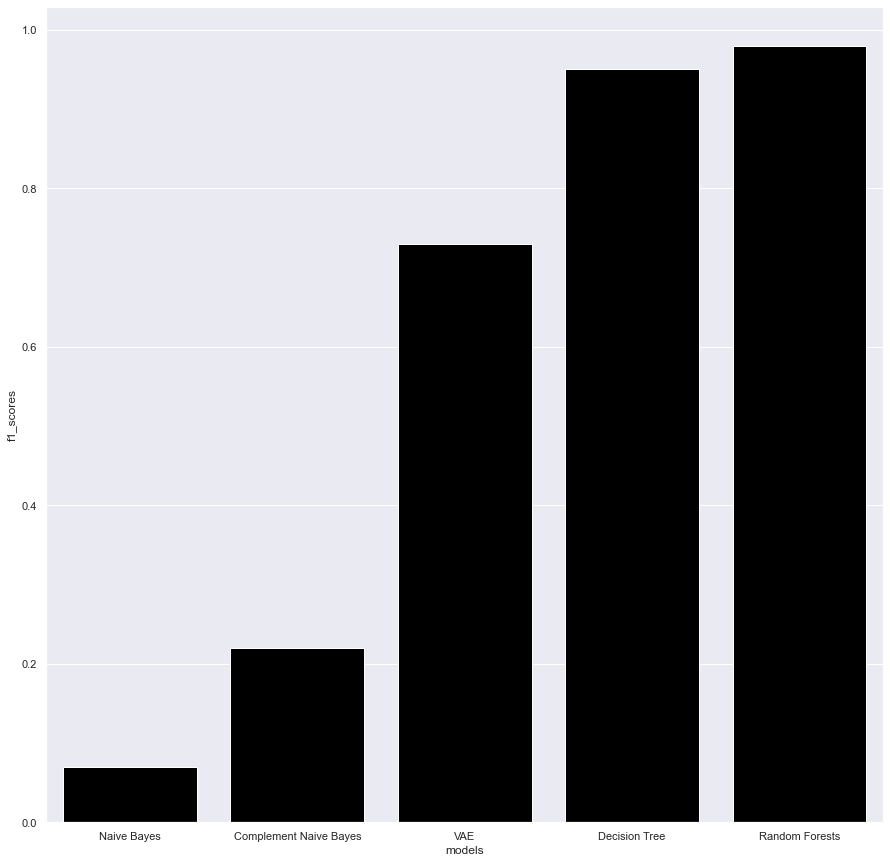
The VAE performed rather disappointingly compared to the other models. It’s AUC was sub-par, and F1-score not satisfactory. Furthermore, its accuracy was low too. However, given the time-constraints of the project – I was not able to fully tune the model. The hyper-parameters were first initialized randomly, then slowly changed in order to improve results. This was a slow and tedious process, and could not be carried out to its full conclusion.

From the Reconstruction Error vs target graph, we can see that Fraudulent posts have a higher Reconstruction error compared to non-fraudulent posts. This indicates some level of difference in the distribution of features by target. Further investigation is required.

## **Champion Model**

All the tree models performed admirably, both HGBT and LGBM performed similarly which was not surprising given that HGBT was inspired by LGBM. Ultimately, the champion model I chose was the Random Forest because:

1. It had similar performances to the gradient boosted models even though LGBM performed the absolute best.
2. It was resource-efficient to run. It didn’t take much computing resources & time to train the model.
3. The model is easily interpretable, and explainable. No advanced concepts need to be covered in order to communicate the model.
4. The model had high accuracy, precision & recall scores. It was able to discriminate between fraudulent & legitimate job posts in a satisfactory manner.

[](https://docs.google.com/spreadsheets/d/1RCJuD-F0fIkCHjKRE1AYYE1KjCVAOtsAgGHnxxMEUjI/copy#gid=209504518)

## **Insights**

1. Whitespaces, consecutive punctuation marks, and number of commas used are good indicators of fraud.
2. Large number of fraudulent posts are focused on the Oil, IT, and healthcare industry.
3. Clickbait Ratio is a highly important feature for most models, and can be used to identify fraudulent posts reliably. Job posts whose description has a clickbait ratio greater than 0.059 are very likely (62%) to be fraudulent.
4. Scammers target vulnerable sections of society by posting lower educational & work experience requirements.
5. Scammers focus on monetary rewards in their job posts – featuring cash rewards & bonuses.
6. Legitimate businesses prioritize reputation, job security and experience.

Based on my findings, I can conclude that the general stereotypes about fake jobs have some element of truth in them. While some fake jobs have very poor grammar, and extraordinarily high compensation – some scammers cleverly disguise their posts in such a way that they are indistinguishable from legitimate advertisements.

# **Governance & Control**

In any analytics project/undertaking, it is important to outline a Governance & Control procedure acknowledging the limitations, downsides, and risks of using the model and analytical framework put forward by the members of the project. Decision-making with the help of decision-support algorithms and models introduces certain blind-spots and biases into an organization’s decision-making framework. A Data and Model Governance policy helps ameliorate the damages arising from these side-effects, as well as helps to mitigate and prevent systematic errors from creeping into an organization’s decision criteria.

Blind-spots in data and modelling may lead to biased decisions especially when those blind-spots are related to rare events/classes. The risks arising from these blind-spots can be classified into several different categories/tiers. In this project, I will be using the Risk Tiers provided by the European Commission where risks are classified into four distinct classes, namely:

## **Risk Matrix**

|  |  |  |
| --- | --- | --- |
| Risk | Risk Level | Action |
| Scammers change tactics to avoid detection | Unacceptable | Conduct feature engineering again, and rebuild models from scratch |
| Model is overfitting (noise vs signal) | High | Rebuild models |
| Model is biased against a certain demographic | Unacceptable | Conduct feature engineering again, and rebuild models from scratch |
| Package/library used in the project is malfunctioning | Unacceptable | This error can go un-noticed for long periods of time. Code-review has to be conducted, and unit tests developed in order to combat this. |
| Features have been engineered incorrectly | Unacceptable | If errors have crept in during the feature engineering stage, then the project has to be redone. |
| Model doesn’t provide enough discriminative power to distinguish between fraud/legitimate job ads | High Risk | Project will have to be redone |
| Data Imputation & manipulation operations are introducing bias/error | Unacceptable | Project will have to be redone |

## **Shifting Targets**

The dataset used for this project was obtained from posts made in 2012-2014. Therefore, variable drift will have occurred. Scammers frequently change their tactics and strategies in order to avoid detection. Therefore, most of the engineered features may need to be adjusted in order to improve detection.

## **Dependency Creep**

I have used several different libraries in this project. They were used for natural language processing tasks, data manipulation, modelling, graphing, visualization, etc. Each of the packages used introduces a certain element of risk into the project. These packages act as black-boxes, and debugging their code requires very specialized skills and expertise. Furthermore, over-time as python & its associated packages are updated, some of the packages used may fall out of fashion & therefore become outdated.

It is important to recognize the immense risk that stems from using external libraries/packages in any project. In order to combat this to some extent, I have documented the packages used in this project. I have also initiated a version control on github so that future users can test, control & verify their versions against mine.

## **Suspicious Results**

I am not entirely convinced by the results of my models. Some of them perform too well given the noisy dataset, and the difficult task. Therefore, I suspect they are over-fitting. However, due to the limited size of this dataset, and the limited availability of fraudulent job posts datasets – it is hard to validate and test my model in a more robust manner. This is why I have stated that creating the models were not my end-goal, but rather a means to an end (gaining insight into fraudster behaviour).

## **Missing Values & Imputation**

Imputation introduces bias and error into datasets. While these operations are not entirely desired, the cost-benefit trade-off must be considered when imputing. If imputation leads to good model development, and good fit statistics, then it can be concluded that imputation was a necessary evil. However, with this dataset – with the lack of an adequate validation set, it is hard to say whether the imputation did more harm than good. It is important to control for every factor into the dataset, and imputation introduces errors which some may find unacceptable.

## **Noise in the dataset**

The nature of this dataset is such that it is hard to separate the signal from the noise. When trying to do so, we are prone to overfitting on noise. In order to combat this, 5-fold Random Search Cross Validation was used. However, since only a max iteration of 60 was used for the complex models due to time-constraints – there is no guarantee that the parameters chosen were the best.

## **Controlling for Drift**

The dataset consists of a total of 73 features which when one-hot encoded unrolls into 128 columns. Each of these features has a distribution/characteristic statistic that will need to be monitored. A tolerance level of +/- 5% should be established, and any deviation greater than that will mean that the models built can no longer be used.

A summary statistic table consisting of the mean, median, mode, etc. has been generated for the engineered variables. New feature values can be compared to those statistics in order to determine whether drift has occurred.

Furthermore, the Term-Document and TF-IDF matrix have 131 columns. When combined with the text/post characteristics features, the combined size amounts to 259. Word-frequency can be monitored by occasionally updating a master TF-IDF matrix based on the latest job postings. Deviations greater than the established 5% limit would lead to a re-examination of the job posts.

# **Recommendations**

## **Real-time Learning System**

The problem of fraud detection is particularly well-suited to real-time analytics, and live learning agents. Because fraudsters tend to change their tactics in order to avoid detection, static models made with past data are bound to become ineffective. In contrast, a continuously learning model able to ingest new data can stay-up-to-date on the latest tactics, and will prove far more effective in detecting fraud.

At the same time, live dashboards and monitoring systems should be built to detect fraud. These dashboards should the latest trend in fraud posts in order to inform analysts, job-seekers, and candidates.

## **Protection Guidelines**

Since scammers target vulnerable and desperate sections of society disproportionately, protection guidelines should be developed & put in place by job advertisement websites and job boards. As per the dataset, the preferred demographic of scammers consists of high-school educated or lower people, looking for clerical/administrative roles, or entering the IT or Oil industry. Since we know which groups the scammers target the most, job posts in these fields/targeting these groups should be put under greater scrutiny.

## **Feedback Platform**

Users need to be given back the power to discuss, collaborate, and flag suspicious job ads. Feedback platforms should be given greater importance by job boards and job ad websites. Peer policing will help improve the quality of job posts on these websites, and also prevent potentially susceptible people from being scammed.

## **Job Ad Dataset**

Lastly, a dataset consisting of job ads (both legitimate and fraudulent) should be made. There is a paucity of such a dataset, and EMSCAD is the only academically accepted dataset in this field. The benefits of creating a massive dataset will greatly outweigh the costs. Just like how the ImageNet database helped kick-start a revolution in Image classification, a job ads dataset may help job-seekers understand how to avoid scams. Intelligent systems to detect fraud can also be developed only when there is a large availability of good quality data.

# **Conclusion**

Fake jobs exploded during the pandemic, and have become a huge nuisance to society. Detecting fraudulent job posts is an inherently difficult task requiring extensive data cleansing, feature engineering and model building. In order to create a good discriminative model, a large amount of data is required which is currently not available. While EMSCAD is a good start for a pilot project, it is not sufficient. Furthermore, since EMSCAD was compiled in the period 2012-2014, the dataset is outdated, and trends/patterns detected in the dataset may have changed.

Scammers target vulnerable sections of society, and prioritize short-term cash rewards & bonuses. They also target fresh graduates, and people looking to start their career. While some fake job ads are obvious, clever scammers disguise their posts in such a way that it becomes undetectable. In such cases, additional indicators will be required to detect fraud. This project aimed to distinguish between legitimate and fraudulent job ads using textual features, but there may be other more viable features which provide better discrimination against fake vs non-fake.

# **References**

CBC News. (2021, September 9). *Fake Job Scams on the rise during pandemic*. Retrieved from cbc.ca: https://www.cbc.ca/news/canada/british-columbia/fake-job-scams-on-rise-during-pandemic-1.6169860

Edmonton Police. (n.d.). *Employment Scams*. Retrieved from edmontonpolice.ca: https://www.edmontonpolice.ca/CrimePrevention/PersonalFamilySafety/Frauds/OnlineScams/EmploymentScams

Liu, J. (2022, June 10). *Americans lost 68 million to job scams*. Retrieved from cnbc.com: https://www.cnbc.com/2022/06/10/americans-lost-68-million-to-job-scams-this-year-here-are-the-most-common-ones.html

Naudé, M., Adebayo, K.J. & Nanda, R. *(2022)A machine learning approach to detecting fraudulent job types.* AI & Soc.doi: 10.1007/s00146-022-01469-0