# AREAS TO IMPROVE

1. Data Collection Pipeline
2. Job Classification Pipeline

**Data Collection Pipeline:** This is a classification problem and due to the unique nature of the problem statement, there are no readily available public datasets of job postings in Nigeria. This leads to the question, how can an automatic data collection pipeline be implemented?

**Ways of implementing Data Collection**

A. Building a Web Scraper

B. Creating a Job Posting Website

**Building a web scraper**

A web scraper is a bot that is built to automatically extract data from a website. To collect Job Posting data, there is a need to first curate the most popular and trusted Job posting websites in Nigeria. This will help in easily extracting data from the sites to build the database. Google Job Posting Search can also be scraped to extract jobs posted in a particular location. It indexes Jobs Posted in Job Posting websites. Facebook Job Posts can also be scraped for jobs posted in a particular location. LinkedIn Jobs are more difficult to scrape because of the company’s anti scraping policy, but it is one of the highest data sources for job postings.

Ways of extracting data from the sites

Due to the legal grey area associated with the use of bots to scrape websites (Privacy breach, Illegal data collection, etc), the best way to continuously scrape a large data source that is continually updated often is to work with the owners of the website to either get access to an API for scraping their sites or get permission to scrape them. Once Permission has been granted, the workflow is as follows:

**A.** Create a database where the scraped data will be stored.

**B.** The scraper will collect data from a particular URL and automatically assign a unique key from the URL (like last 3 characters) from where the data was extracted and stored in the database. The unique key is essential to avoid data duplicates and hence only recent and updated data can be scraped.

**C.** Anytime the scraper encounters a new URL, it will generate the key and check if the key is in the database, if yes then it won't store the data.

**D.** The Web Scraper is built to iterate at a frequency as not to overload the server.

Libraries needed

1. Selenium: This mimics how a human will interact with a browser and is one of the easiest libraries to learn. It can be installed in the anaconda prompt by running the "pip install selenium" command.
2. BeautifulSoup4: Beautiful Soup is a library that makes it easy to scrape information from web pages. It is also easy to learn and one of the most popular libraries for scraping websites. It can be installed in the anaconda prompt by running the "pip install beautifulsoup4 " command.
3. Scrapy: Scrapy is an open-source framework that is both a general-purpose web scraper and also uses API to extract data. It can be installed in the anaconda prompt by running the "pip install scrapy" command.

**Create a Job Posting Site:** This is an over the top solution but can be useful in the long run. This solution might come in handy if the web scraping process encounters issues in terms of permission and legality. Although expensive, it is advisable to run a job posting website alongside a job classification website, this will give an unlimited access to job’s database if the popularity of the site increases.

**Job Classification Pipeline**

The summary of the process flow is as follows

1. Obtain an external online recruitment fraud dataset

2. Clean and Preprocess the data

3. Use Support vector machine to select relevant features

4. Train the model using the random forest algorithm

5. Use the trained model to predict the database collated from the data collection pipeline as a means of testing the accuracy of the model (the assumption being that the database is filled with real jobs since they were scraped from real job sites)

6. For low accuracies, use a 2-factor authentication method to confirm the authenticity of a job, meaning the job posting prediction should be real and the job posting should be in the database.

7. Change the form that will receive user inputs to reflect the features selected in the training process

8. Retrain the model, this time include all the data in the database to improve performance.

Obtain the data.

1. The data is an online recruitment fraud CSV data gotten from

[http://emscad.samos.aegean.gr](http://emscad.samos.aegean.gr/).

The data consists of 17, 860 rows and 18 columns. It is a highly imbalanced dataset with 17,014 real jobs and 866 fake jobs. This data will be trained in to help in the prediction of a job posting

Clean and pre-process the data

The code for cleaning and preprocessing the data can be seen in this github repository

<https://github.com/chidibede/job-prediction>. Basically, all missing values should be handled, HTML tags removed, trailing white spaces accounted for and the text converted to vectors, this will make the data ready for training and feature selection. The libraries needed include pandas and numpy

Feature selection: to remove unnecessary features the impactful features will be selected by using SVM, from my trial 4 features were important and this had a great bearing on the authenticity of a job posting. One of the features is the presence of a company logo. This means that once a company logo is present, it heavily impacts on the authenticity of a job posting. This leads to the need to add this column in the database when collecting job posting data in the data collection pipeline stated at the beginning of this research. The library needed is the Sklearn Machine Learning Library.

Other solutions

Real job prediction based on location

If no location is entered, the job prediction should be fake but once a location is entered, the location is verified by using Google's map API which will confirm If the job is real or not.