Milestone 3 - Report

Project Title : Fake Vs Real Job Posting Analysis

Course - SP20: CMPE-255 Sec 01 - Data Mining

Project Group - **9**

Group Members -
Rutuja Hasurkar (014547793)
Sudarshan Aithal (013638703)
Shivan Desai (010279646)
Jasmine Akkal (013773825)

Table of Contents

Abstract
1) Introduction/Background
2) Pre processing and Exploratory Data Analysis
3) Method Description
    3.1) Random Forest Classifier
    3.2) TF-IDF
    3.3) NLP ulmfit
    3.4) XGBoost
4) Preliminary Results
5) Experiment/Analysis
6) Comparisons
7) Conclusion
8) Repository
9) References

Abstract

With the increase in the number of job applications and job postings, comes along the possibility of misusing this information and the platform to post this information. After research and observation of the various job posting portals it has been observed that the job posting sites are filled with fake job postings which may lead to the applicants information being misused. In order to know and understand these job postings we implement Data mining techniques and algorithms on the famous data set provided by the University of the Aegean on the kaggle platform. In this project, we study the dataset which consists of 18 columns and around 17 thousand records. We learn features of the dataset by applying Exploratory Data Analysis (EDA) and then do Data preprocessing to handle null values and missing values. Further we implement different models on the dataset to classify the data into real and fake job postings. The primary motive to come up with this project is to help job seekers to avoid the fake job postings which can put their privacy at stake by exposing their personal details in unsafe hands.

The main approach of this project is to apply classification algorithms to classify the data, that is, the job posting into mainly two categories, such as "Real" or "Fake" job posting. The First approach towards implementation of a model is Random Forest Classifier, which focuses on the non descriptive attributes like location, industry, etc. Going further we make use of Pearson's correlation and Pearson's Chi Square test to find the dependent variable and feature selection. TF - IDF and pretrained NLP algorithms like ULMFit will be used to develop models to work on descriptive data to find the pattern of words of a fake job posting and test them out on a training set. We present the training and testing accuracies of the various models implemented to support our approach.

1. Introduction / Background

Description :

With ease of access to technologies, the number of jobs and number of job applicants have been significantly increasing in recent years. With this comes fraudulent techniques which misuses the dire situation of applicants by stealing their data for their malicious uses. We are aiming at developing a model to find fake job postings from groups of job postings. The data is provided by The University of the Aegean. This can help a number of job seekers to avoid the fake job postings which can actually expose their personal details in unsafe hands.

Approach :

Main approach of this project is to use data mining techniques to prepare the data for respective models. This is a classification task. So our model will be based on Apriori Association and random forest classification which concentrates on non descriptive attributes like, location, industry, Title, job_id etc. We will be using Data Mining techniques to process the Job Postings data. Using Pearson's correlation and Pearson's Chi Square test to find the dependent variable and feature selection. TF - IDF and pretrained NLP algorithms like ULMFit will be used to develop models to work on descriptive data to find the pattern of words of a fake job posting and test them out on a training set.

Our approach differs from the ones already implemented is on the basis that We will be concentrating on using TF-IDF on selected features of the data, using ULMFit algorithm to analyze the texts, Classification models like Random forest and Apriori association or if time favors, a hybrid model.

Related Work :

The previous work on this dataset includes implementation of NLP algorithms using FastAI, BERT analysis and TF-IDF. Also, due to the high imbalance of real job vs fake jobs in the dataset, people have also used various oversampling and undersampling techniques to make the data more balanced for a more accurate prediction.

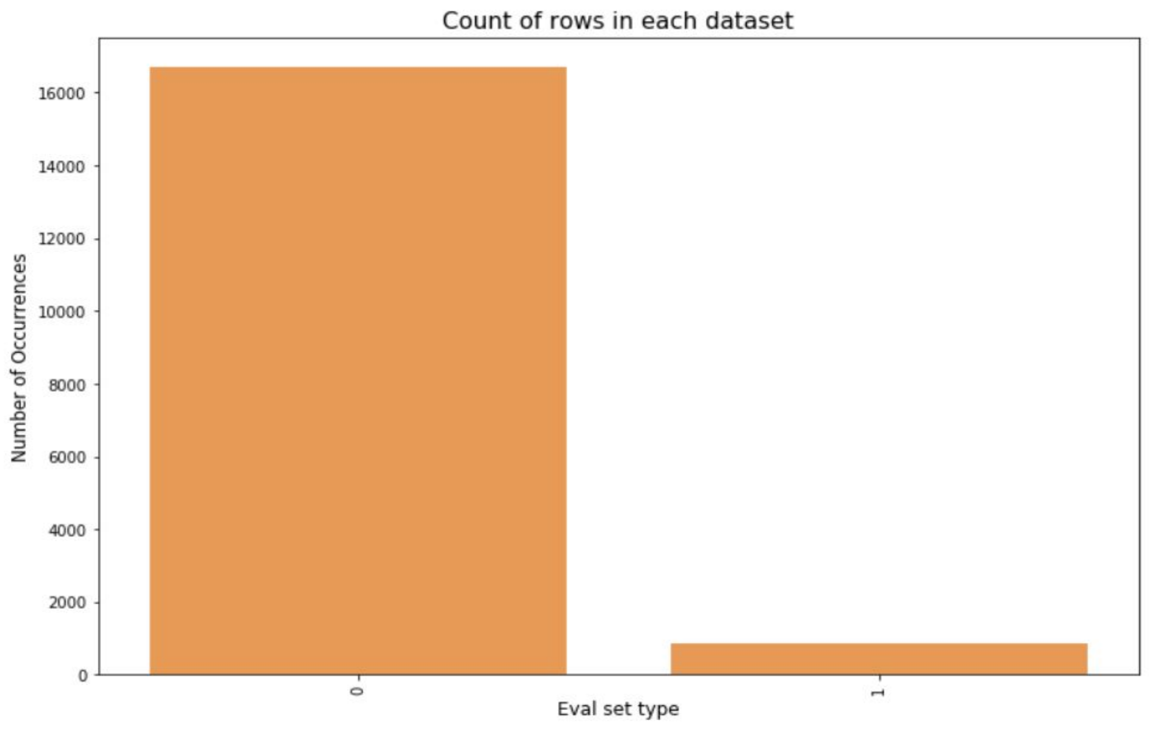
Dataset :

- Dataset is provided by The University of the Aegean.
- It consists of 18 columns.
    - Job_id : Unique Job ID
    - Title : The title of the job ad entry.
    - Location : Geographical location of the job ad.
    - Department : Corporate department (e.g. sales).
    - Salary_range : Indicative salary range (e.g. $50,000-$60,000)
    - Company_profile : A brief company description.
    - Description : The details description of the job ad.
    - Requirements : Enlisted requirements for the job opening.
    - Benefits : Enlisted offered benefits by the employer.
    - Telecommuting : True for telecommuting positions.
    - Has_company_logo : True if company logo is present.
    - Has_questions : True if screening questions are present.
    - Employment_type : Full-type, Part-time, Contract, etc.
    - Required_experience : Executive, Entry level, Intern, etc.
    - Required_education : Doctorate, Master’s Degree, Bachelor, etc.
    - Industry : Automotive, IT, Health care, Real estate, etc.
    - Function : Consulting, Engineering, Research, Sales etc.
    - Fraudulent : target - Classification attribute.

2. Pre processing and Exploratory Data Analysis

Data pre processing:

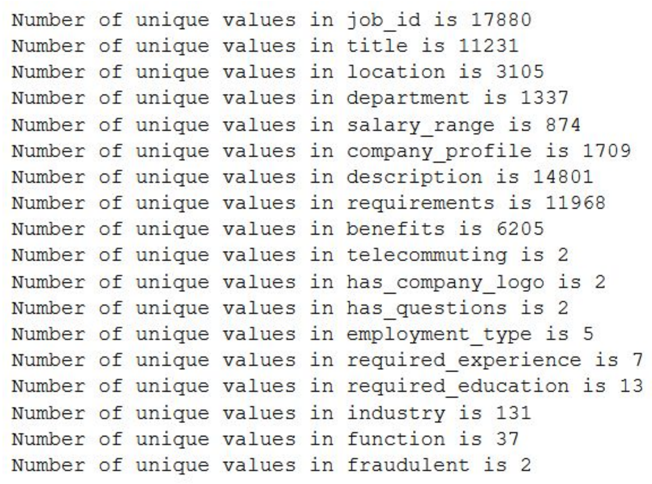
- The dataset was provided by the Laboratory of Information & Communication Systems Security Of The University of the Aegean.
- The dataset consists of 18k job descriptions with 800 fake job descriptions.



- There are 18 columns in the dataset
    - Job_id : Unique Job ID
    - Title : The title of the job ad entry.
    - Location : Geographical location of the job ad.
    - Department : Corporate department (e.g. sales).
    - Salary_range : Indicative salary range (e.g. $50,000-$60,000)
    - Company_profile : A brief company description.
    - Description : The details description of the job ad.
    - Requirements : Enlisted requirements for the job opening.
    - Benefits : Enlisted offered benefits by the employer.
    - Telecommuting : True for telecommuting positions.
    - Has_company_logo : True if company logo is present.

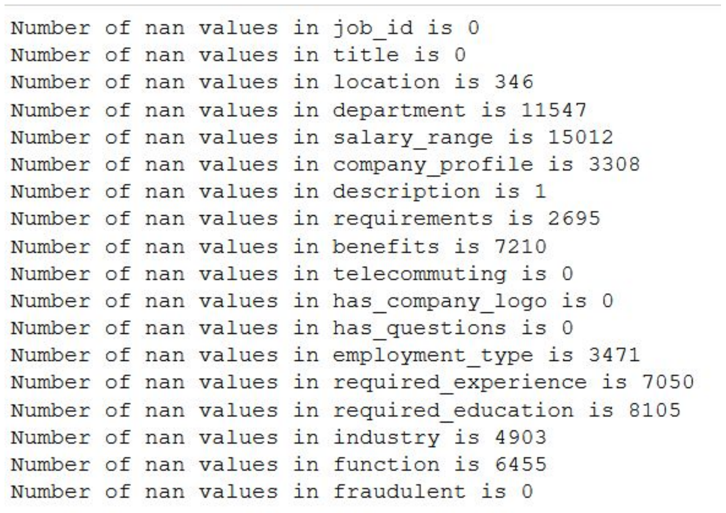
- Has_questions : True if screening questions are present.
- Employment_type : Full-type, Part-time, Contract, etc.
- Required_experience : Executive, Entry level, Intern, etc.
- Required_education : Doctorate, Master’s Degree, Bachelor, etc.
- Industry : Automotive, IT, Health care, Real estate, etc.
- Function : Consulting, Engineering, Research, Sales etc.
- Fraudulent : target - Classification attribute.

- The number of unique values in a column determines the number of states that feature can take. The higher the uniqueness, the more complex the algorithm should be to adapt varying states. Following are the number of unique values each column had.



```
Number of unique values in job_id is 17880
Number of unique values in title is 11231
Number of unique values in location is 3105
Number of unique values in department is 1337
Number of unique values in salary_range is 874
Number of unique values in company_profile is 1709
Number of unique values in description is 14801
Number of unique values in requirements is 11968
Number of unique values in benefits is 6205
Number of unique values in telecommuting is 2
Number of unique values in has_company_logo is 2
Number of unique values in has_questions is 2
Number of unique values in employment_type is 5
Number of unique values in required_experience is 7
Number of unique values in required_education is 13
Number of unique values in industry is 131
Number of unique values in function is 37
Number of unique values in fraudulent is 2
```

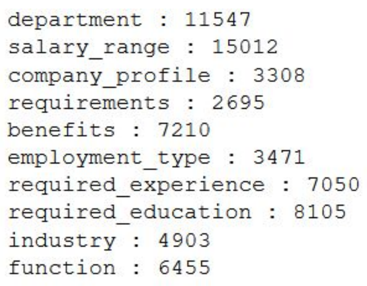
- The multiple columns of the dataset consisted of null values.



```
Number of nan values in job_id is 0
Number of nan values in title is 0
Number of nan values in location is 346
Number of nan values in department is 11547
Number of nan values in salary_range is 15012
Number of nan values in company_profile is 3308
Number of nan values in description is 1
Number of nan values in requirements is 2695
Number of nan values in benefits is 7210
Number of nan values in telecommuting is 0
Number of nan values in has_company_logo is 0
Number of nan values in has_questions is 0
Number of nan values in employment_type is 3471
Number of nan values in required_experience is 7050
Number of nan values in required_education is 8105
Number of nan values in industry is 4903
Number of nan values in function is 6455
Number of nan values in fraudulent is 0
```

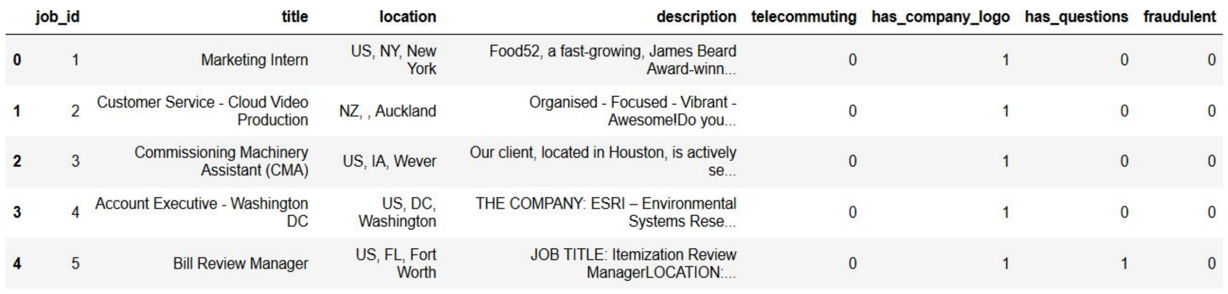
Our task was to eliminate columns with at least 10% of Null value depending on the algorithms' focus. As sparse columns will not contribute much to the learning.

- Following columns had more than 10% of Null values.



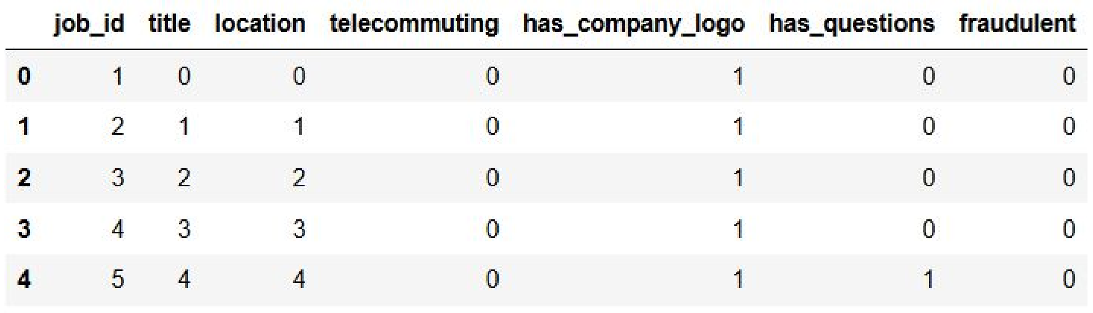
```
department : 11547
salary_range : 15012
company_profile : 3308
requirements : 2695
benefits : 7210
employment_type : 3471
required_experience : 7050
required_education : 8105
industry : 4903
function : 6455
```

- A peek into the dataset after the dataset columns with more than 10% null values were removed.



	job_id	title	location	description	telecommuting	has_company_logo	has_questions	fraudulent
0	1	Marketing Intern	US, NY, New York	Food52, a fast-growing, James Beard Award-winn...	0	1	0	0
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Organised - Focused - Vibrant - Awesome!Do you...	0	1	0	0
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	Our client, located in Houston, is actively se...	0	1	0	0
3	4	Account Executive - Washington DC	US, DC, Washington	THE COMPANY: ESRI – Environmental Systems Rese...	0	1	0	0
4	5	Bill Review Manager	US, FL, Fort Worth	JOB TITLE: Itemization Review ManagerLOCATION:...	0	1	1	0

- We used tokenization to convert non-numerical attributes to numerical attributes that can be further used by the algorithm to classify. For Non NLP classifier algorithms, using descriptive information for learning may not be an easy task as to take tokenizing Descriptive information may lead to all unique numbers and learning can be hard.

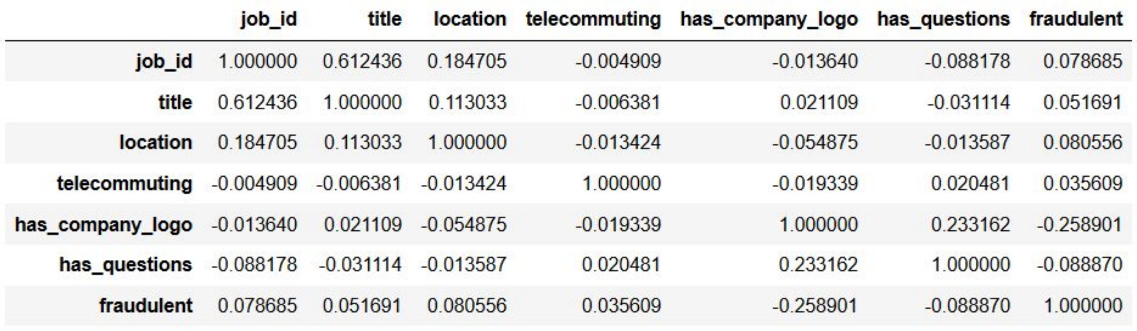


	job_id	title	location	telecommuting	has_company_logo	has_questions	fraudulent
0	1	0	0	0	1	0	0
1	2	1	1	0	1	0	0
2	3	2	2	0	1	0	0
3	4	3	3	0	1	0	0
4	5	4	4	0	1	1	0

Depending on the algorithm, preprocessing of the data can be varied to yield efficient learning

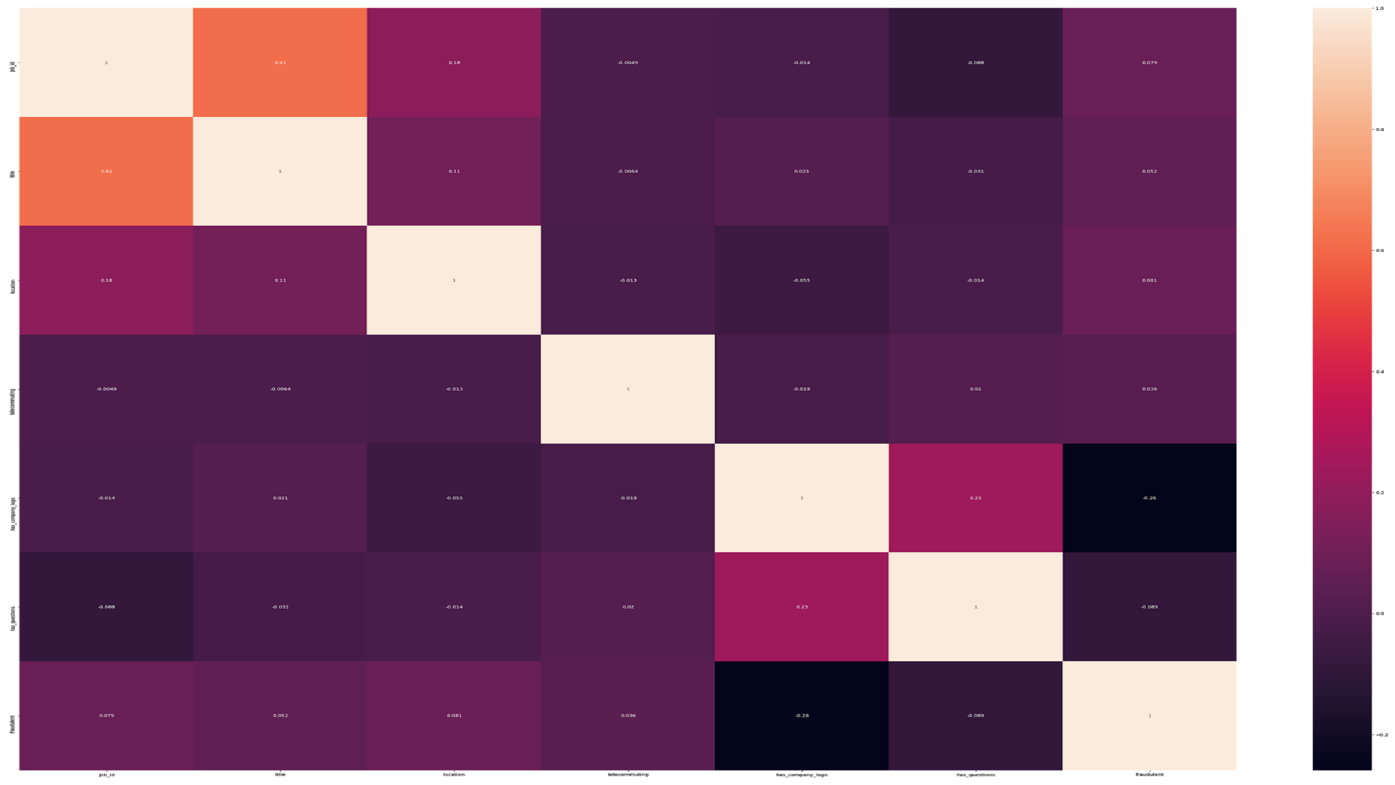
- To determine dependent attributes that contribute to the final prediction of a fake or real job, we are using Pearson’s Correlation and Pearson’s Chi Square Test.
  - Pearson’s correlation is used to determine a coefficient between two attributes which defines if those two attributes are linearly dependent or not. If the
    Coefficient value is 0, Then the two variables are independent of each other
    Coefficient value is 1, the two variables are dependent of each other
  - Pearson’s Chi Square Test is also used to define linear dependency between two attributes. By defining a null hypothesis, we determine the observed frequency of an attribute by using another attribute and compare it with the original value. We get a “*p value*” that determines if the hypothesis chosen to correlate two attributes should be accepted or rejected.
  - A good practice is to select p-value and pearson’s correlation coefficient to be at least > 0.05

- Pearson’s correlation of factorized, description removed, greater than 10% null valued column removed data set is:

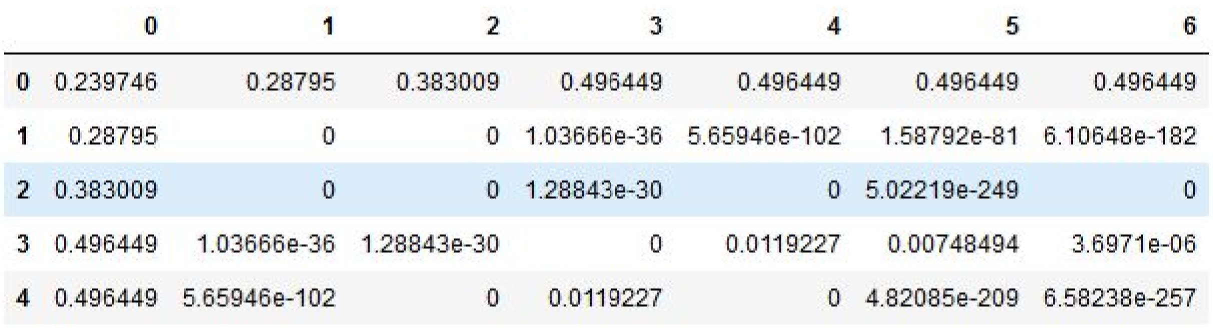


	job_id	title	location	telecommuting	has_company_logo	has_questions	fraudulent
job_id	1.000000	0.612436	0.184705	-0.004909	-0.013640	-0.088178	0.078685
title	0.612436	1.000000	0.113033	-0.006381	0.021109	-0.031114	0.051691
location	0.184705	0.113033	1.000000	-0.013424	-0.054875	-0.013587	0.080556
telecommuting	-0.004909	-0.006381	-0.013424	1.000000	-0.019339	0.020481	0.035609
has_company_logo	-0.013640	0.021109	-0.054875	-0.019339	1.000000	0.233162	-0.258901
has_questions	-0.088178	-0.031114	-0.013587	0.020481	0.233162	1.000000	-0.088870
fraudulent	0.078685	0.051691	0.080556	0.035609	-0.258901	-0.088870	1.000000

- Heatmap of Pearson’s correlation.

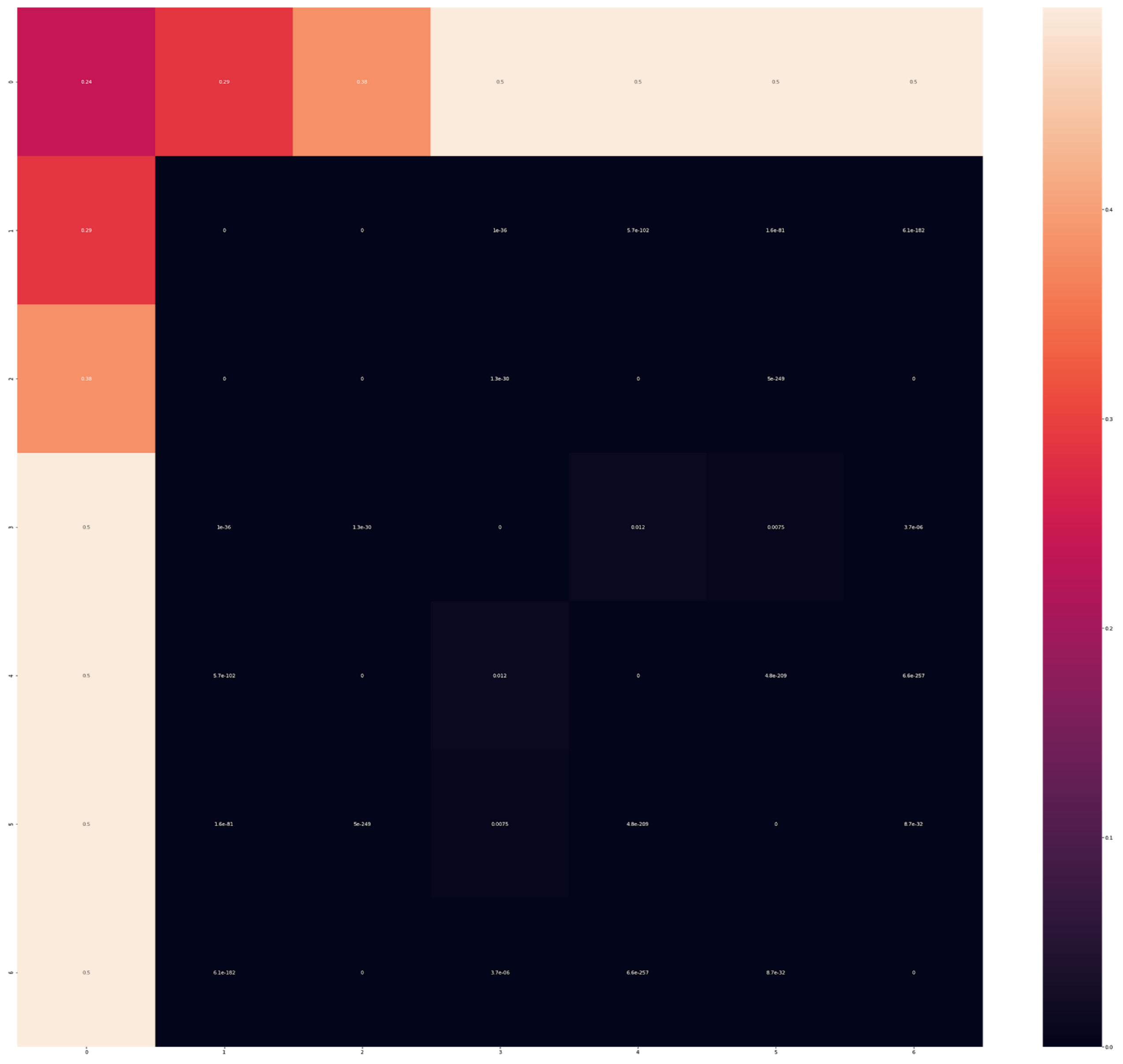


- P values for factorized, description removed, greater than 10% null valued column removed data set is:



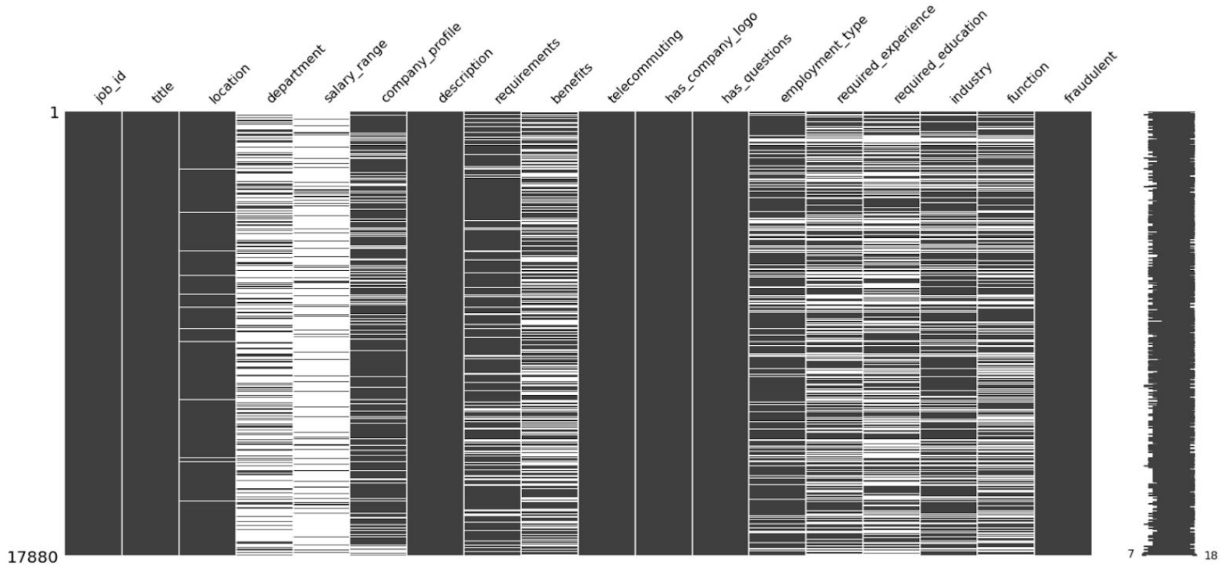
	0	1	2	3	4	5	6
0	0.239746	0.28795	0.383009	0.496449	0.496449	0.496449	0.496449
1	0.28795	0	0	1.03666e-36	5.65946e-102	1.58792e-81	6.10648e-182
2	0.383009	0	0	1.28843e-30	0	5.02219e-249	0
3	0.496449	1.03666e-36	1.28843e-30	0	0.0119227	0.00748494	3.6971e-06
4	0.496449	5.65946e-102	0	0.0119227	0	4.82085e-209	6.58238e-257

- Heatmap for P values.

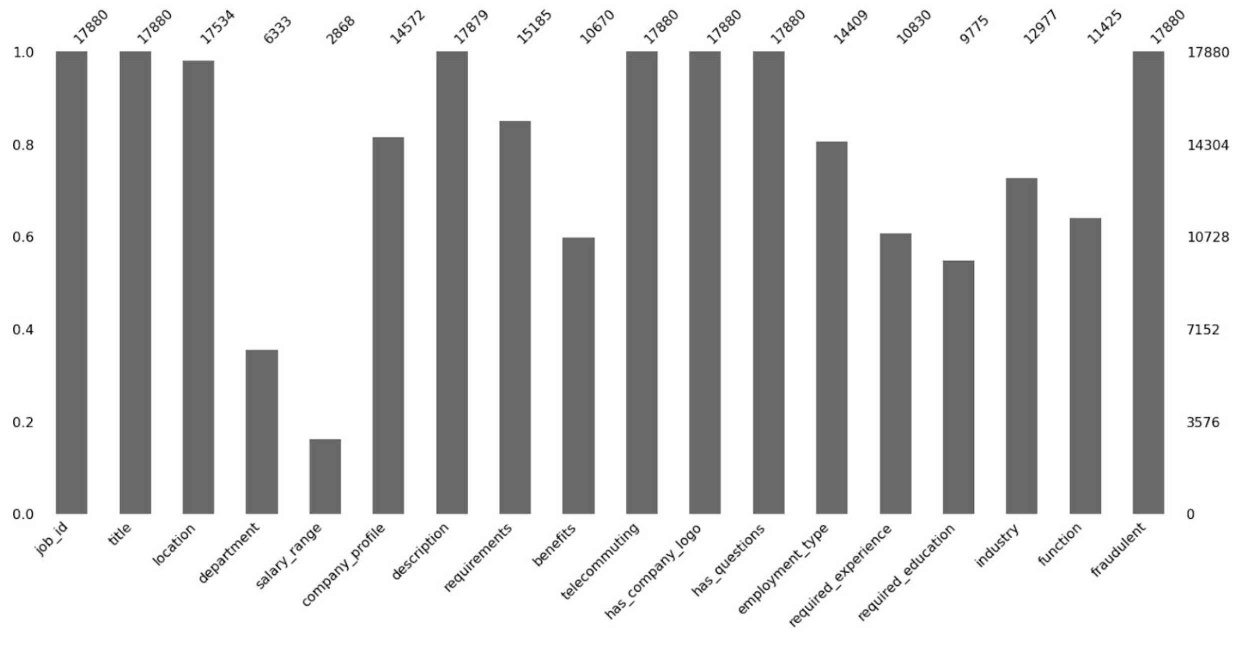


Exploratory Data Analysis (EDA):

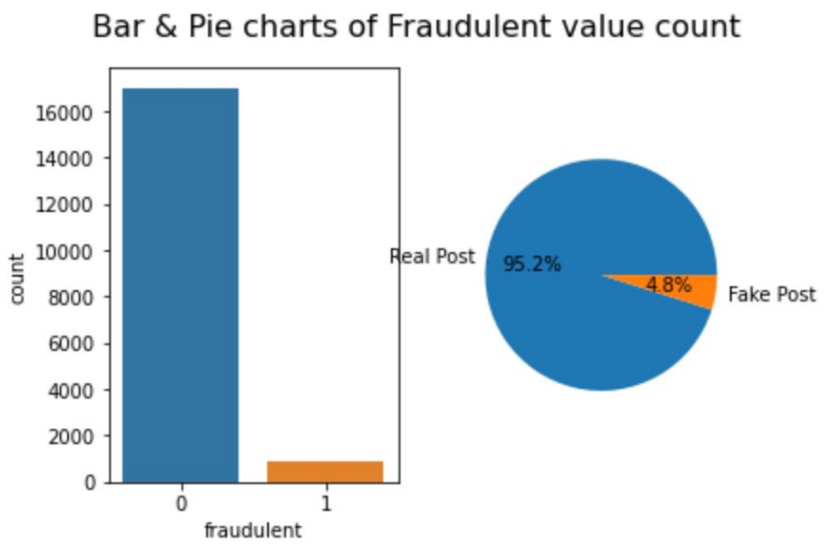
- Visualizing the missing values in the dataset. We use missingno library to plot and get the visual representation of the records and columns representing the cells which have missing values. In the following plot the black are filled records whereas the white spaces represent the missing field in the dataset.



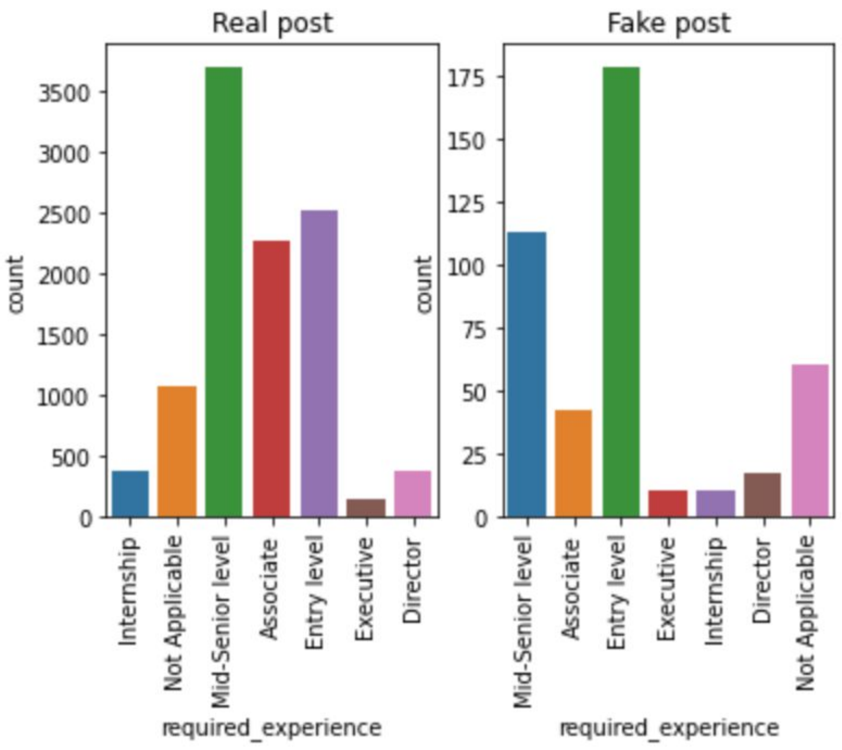
- Plotting a bar graph for representing the missing value count in the dataset.



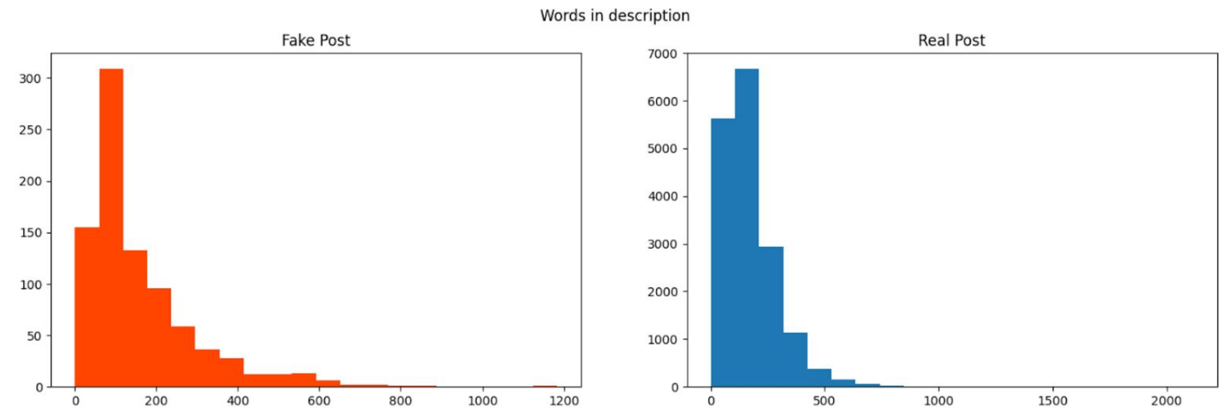
- Graphical representation of the count value of the target field. For our dataset we have the column “fraudulent” as our target field. We use a bar chart and pie chart to display the count of this column.



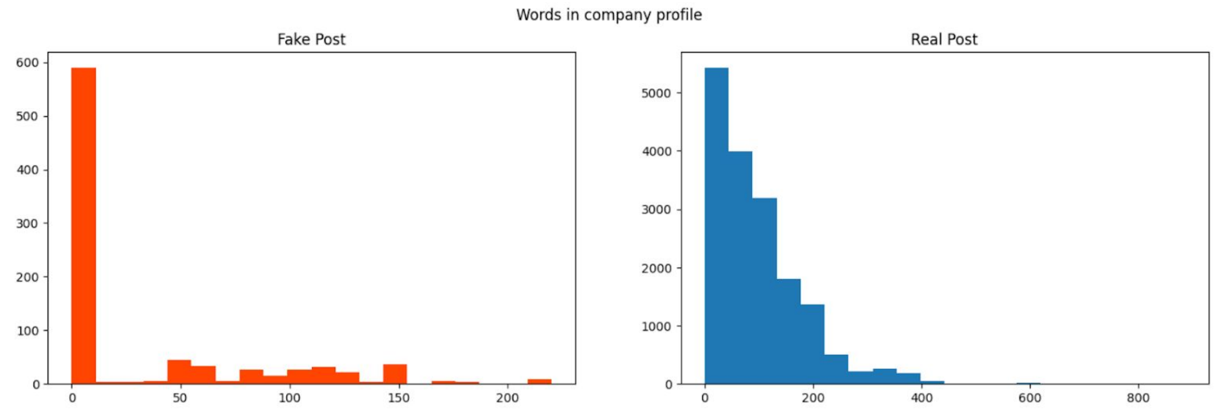
- Analysing the column of the dataset “requird_experience”. We are doing the following analysis on the training dataset.



- The job description plays a major role in understanding if the job post is real or fake. So we analyse the column “description” of the dataset to count the number of words. We observe how the trend is with respect to the word count and real / fake job posting.



- Word count analysis on the column “company_profile”.



3. Method Description:

3.1) Random Forest Classifier

Random Forest Classifier is an ensemble learning algorithm. Ensembled algorithms can be explained as those which implements a combination of more than a single algorithm of similar or different types for classification purpose. Random FOrest Classifier can be explained to be an algorithm which creates a set of decision trees by selection of random subsets of the training dataset. It then accumulates the results from different smaller decision trees to decide the final class of the test object.

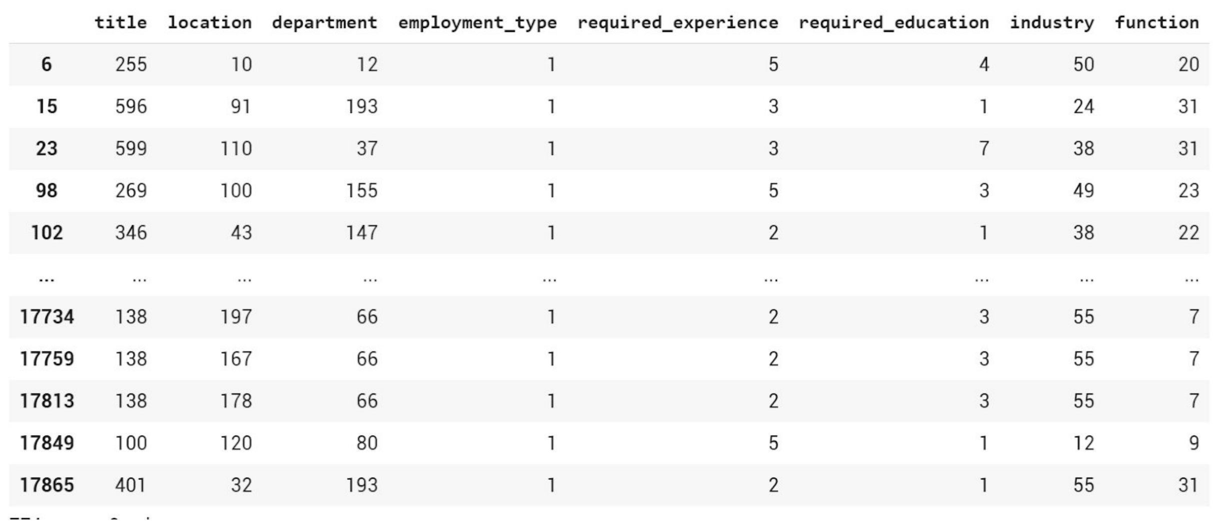
Suppose we have a training dataset as : [A1, A2, A3, A4, A5] which are labelled as [L1, L2, L3, L4, L5]. The random forest classifier divides the training set into smaller subsets like [A1, A2, A3], [A2, A4, A5], [A1, A3, A5], [A3, A4, A5] and likewise. The algorithm calculates the accuracy for individual subsets and then aggregates the results of the subsets to finally classify the data from the test dataset. [9.2]

Random forest classifier models are used over decision trees because a single decision tree may be prone to noise. But, with results from multiple decision trees aggregated may reduce the effect of the noise, thus giving more accurate results and efficiency on large data bases. The parameters used by the random forest classifier are the total number of trees to be generated and decision tree related parameters like minimum split criteria.

The reason behind selecting the random forest classifier model for the job posting analysis and fake job detection dataset is that this dataset has around 18K records over 18 columns, which comprises missing data. Random forest classifier model is well known for being able to handle large input variables, it gives estimates of what variables are important in the classification. Further, it has an effective method for estimating the missing data and maintains accuracy when a large proportion of the data are missing.

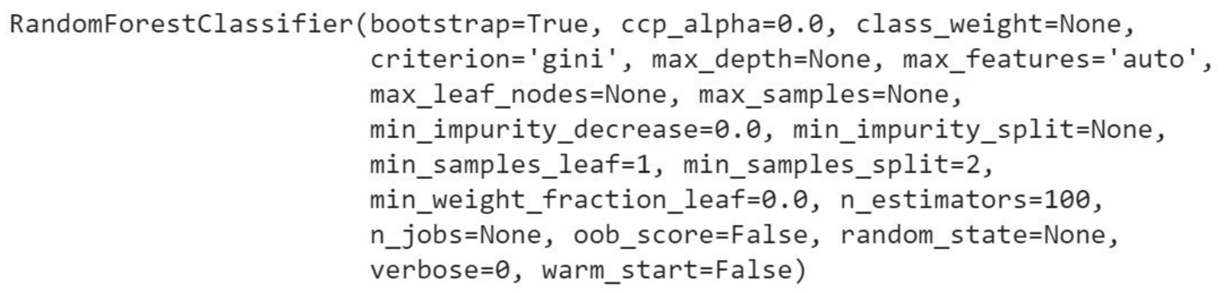
Implementation :

We implement the model on the dataset using the sklearn library's RandomForestClassifier() function. This model implementation focuses on the non descriptive attributes/ columns of the dataset. Use them as categorical attributes.



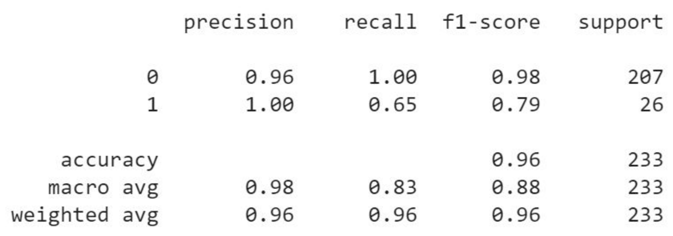
	title	location	department	employment_type	required_experience	required_education	industry	function
6	255	10	12	1	5	4	50	20
15	596	91	193	1	3	1	24	31
23	599	110	37	1	3	7	38	31
98	269	100	155	1	5	3	49	23
102	346	43	147	1	2	1	38	22
...	...	...	...	...	...	...	...	...
17734	138	197	66	1	2	3	55	7
17759	138	167	66	1	2	3	55	7
17813	138	178	66	1	2	3	55	7
17849	100	120	80	1	5	1	12	9
17865	401	32	193	1	2	1	55	31

The reason that makes this RandomForestClassifier different from the existing model implementation on this dataset is that we fine tune the model parameters to best fit the results and accuracies. The parameters such as criterion, max_features, min_samples_fit, n_estimators and the values to these parameters in the argument were closely observed to find a combination of parameter-values to gain the most accurate accuracy.



```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Results :
Accuracy : 96%


```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       207
           1       1.00      0.65      0.79        26

    accuracy                           0.96       233
   macro avg       0.98      0.83      0.88       233
weighted avg       0.96      0.96      0.96       233
```

3.2) TF-IDF

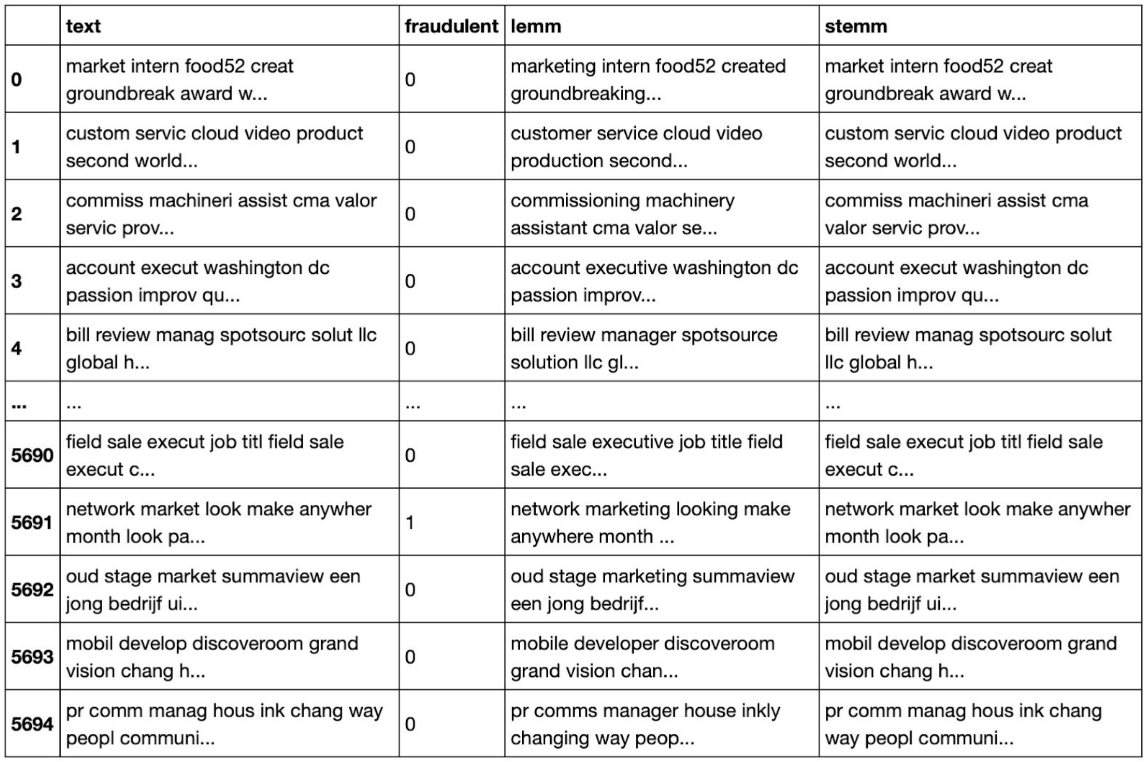
TF-IDF is a numeric statistic that helps determine how important a word is for a document in a document or corpus. It has two parts, namely term frequency and inverse document frequency. A lot of times, these parts might be used seperately in various situations. However, TF-IDF is the combination of these two statistics.

Term Frequency: To find the term frequency of a phrase or a query, we first eliminate all the documents that do not contain the word. This usually eliminates many documents. Following this, we count the number of times each term of the phrase occurs in each document that contains the words. This process basically helps us determine the ‘weight’ of each word in the phrase.

Inverse document frequency: Term frequency incorrectly gives a lot of weight to words that occur more frequently, but might be very less significant in the actual analysis, and gives less importance to meaningful terms that might have more significance. Such extremely frequent words (called ‘stop words’ eg: ‘the’) should first be removed before we apply TF-IDF. However, to diminish the weight of other words that might occur very frequently and increase the weights of those words that occur rarely, an inverse document frequency is incorporated. Inverse document frequency is a measure of how much information a word provides for the analysis.
To calculate the inverse document frequency, we first divide the total number of documents by the number of documents that contain the term, followed by calculating the logarithm of the value we obtain.

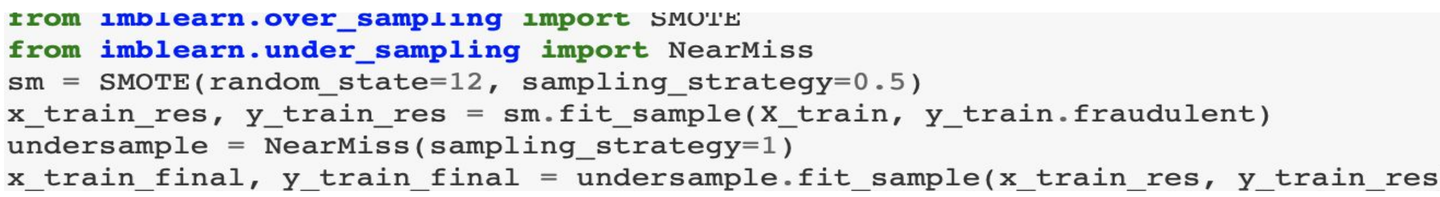
TF-IDF is the product of TF and IDF.

For our analysis, since TF-IDF vectorization only works on descriptive columns, we created a data set that is a combination of all the descriptive columns of the original dataset. Following this, we cleaned the dataset by removing punctuations, and all other non-alphanumeric characters from the column. We also applied lemmetization and stemming to make the data more appropriate for the model.



	text	fraudulent	lemm	stemm
0	market intern food52 creat groundbreak award w...	0	marketing intern food52 created groundbreaking...	market intern food52 creat groundbreak award w...
1	custom servic cloud video product second world...	0	customer service cloud video production second...	custom servic cloud video product second world...
2	commiss machineri assist cma valor servic prov...	0	commissioning machinery assistant cma valor se...	commiss machineri assist cma valor servic prov...
3	account execut washington dc passion improv qu...	0	account executive washington dc passion improv...	account execut washington dc passion improv qu...
4	bill review manag spotsourc solut llc global h...	0	bill review manager spotsource solution llc gl...	bill review manag spotsourc solut llc global h...
...	...	...	...	...
5690	field sale execut job titl field sale execut c...	0	field sale executive job title field sale exec...	field sale execut job titl field sale execut c...
5691	network market look make anywher month look pa...	1	network marketing looking make anywhere month ...	network market look make anywher month look pa...
5692	oud stage market summaview een jong bedrijf ui...	0	oud stage marketing summaview een jong bedrijf...	oud stage market summaview een jong bedrijf ui...
5693	mobil develop discoveroom grand vision chang h...	0	mobile developer discoveroom grand vision chan...	mobil develop discoveroom grand vision chang h...
5694	pr comm manag hous ink chang way peopl communi...	0	pr comms manager house inkly changing way peop...	pr comm manag hous ink chang way peopl communi...

Then, we vectorized the text column using TF-IDF vectorizer, and split the dataset into training and testing datasets.
Since our original dataset was heavily imbalanced, we oversamples the minority class to about 0.5 * size of the majority class in the training dataset using SMOTE. We also undersampled the majority class to equal the size of the minority class in the training dataset using NearMiss undersampling.



```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss
sm = SMOTE(random_state=12, sampling_strategy=0.5)
x_train_res, y_train_res = sm.fit_sample(X_train, y_train.fraudulent)
undersample = NearMiss(sampling_strategy=1)
x_train_final, y_train_final = undersample.fit_sample(x_train_res, y_train_res
```

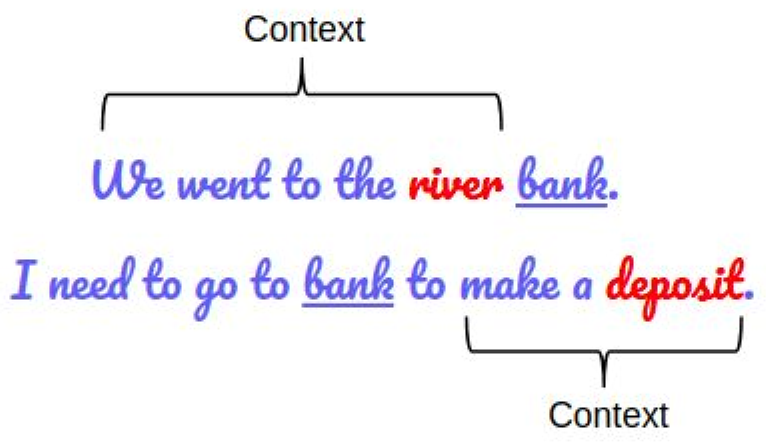
Finally, we used the TF-IDF vector to analyze the data using various models like Logistic regression, KNN, Naive-Bayes, etc.

In the results section we can see that the accuracy of the models like DTrees and KNN is slightly more in the imbalanced dataset, but overall, the average accuracy of the models is higher for the balanced dataset when analyzed along with the TF-IDF vectorizer.
We initially assumed that the accuracy won’t be very high for the balanced dataset, but we aimed to maximize the accuracy using various methods for the balanced dataset. However, we were surprised to see that the average accuracy for the models using the balanced dataset was higher than the unbalanced dataset.

3.3) Natural-Language Programming Countvectorizer and BERT:

Count vectorizer Model : Predictive Modeling requires text data to be specially prepared before we can start working on it. The process of tokenization involves removing words from the text and is required to be carried out. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction (or vectorization).The model is simple in that it throws away all of the order information in the words and focuses on the occurrence of words in a document. This can be done by assigning each word a unique number. Then any document we see can be encoded as a fixed-length vector with the length of the vocabulary of known words. The value in each position in the vector could be filled with a count or frequency of each word in the encoded document. In our code we have already derived accuracy for 5 folds for the count vectorizer model.

BERT: Bidirectional Encoder Representations from Transformers is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context and that pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.
BERT is actually pre-trained on a large corpus of unlabelled text including the entire wikipedia data and the book corpus. This entire data set on which the pre-training has been done is the beauty of the BERT and it is a very huge data we are talking about makes it more swift as a model.
BERT is ‘deeply bidirectional’ which means it learns from both the left and right side of the token context during its training phase.


Context
We went to the river bank.
I need to go to bank to make a deposit.
Context

BERT captures both the left and right context

If we try to predict the nature of the word “bank” by only taking either the left or the right context, then we will be making an error in at least one of the two given examples.

One way to deal with this is to consider both the left and the right context before making a prediction. That’s exactly what BERT does! We will see later in the article how this is achieved.

And finally, the most impressive aspect of BERT. We can fine-tune it by adding just a couple of additional output layers to create state-of-the-art models for a variety of NLP tasks.

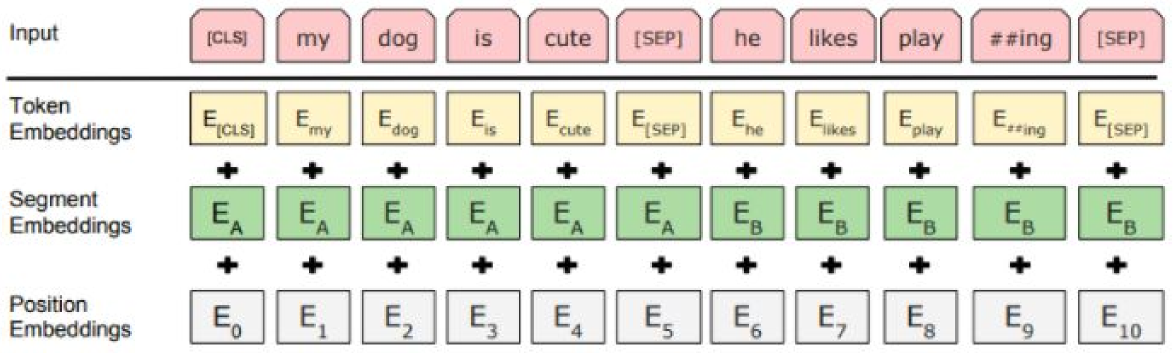
BERT’s Architecture

The BERT architecture builds on top of the Transformer. We currently have two variants available:

- BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
- BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters

Text Preprocessing

- Position Embeddings: BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture “sequence” or “order” information
- Segment Embeddings: BERT can also take sentence pairs as inputs for tasks (Question-Answering). That’s why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB)
- Token Embeddings: These are the embeddings learned for the specific token from the WordPiece token vocabulary


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

These combinations of preprocessing steps make BERT so versatile. This implies that without making any major change in the model’s architecture, we can easily train it on multiple kinds of NLP tasks.

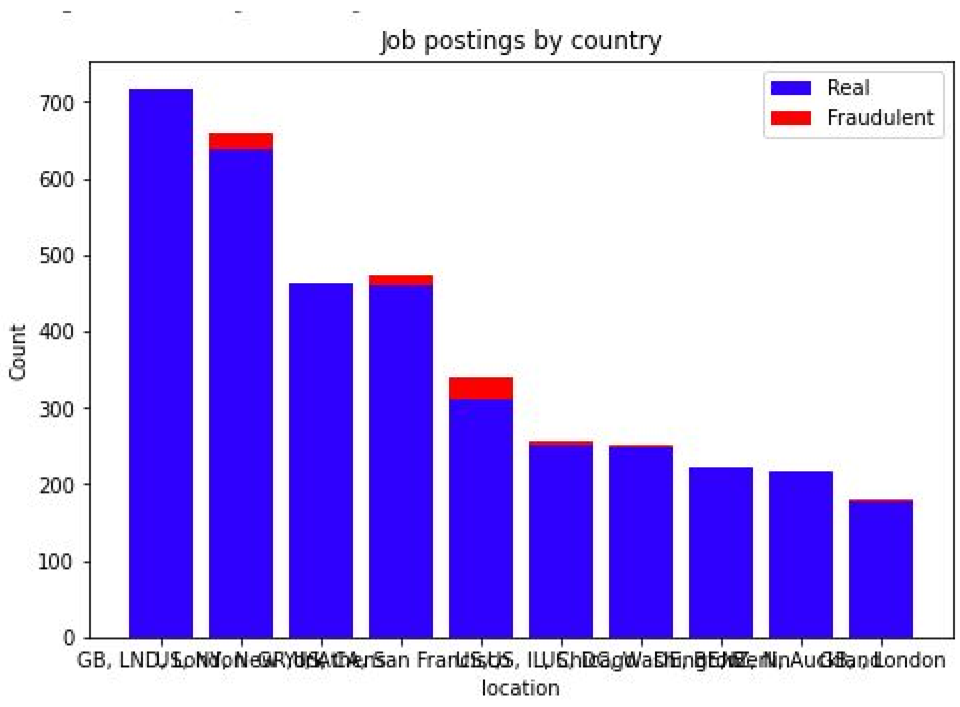
Pre-training Tasks

BERT is pre-trained on two NLP tasks:

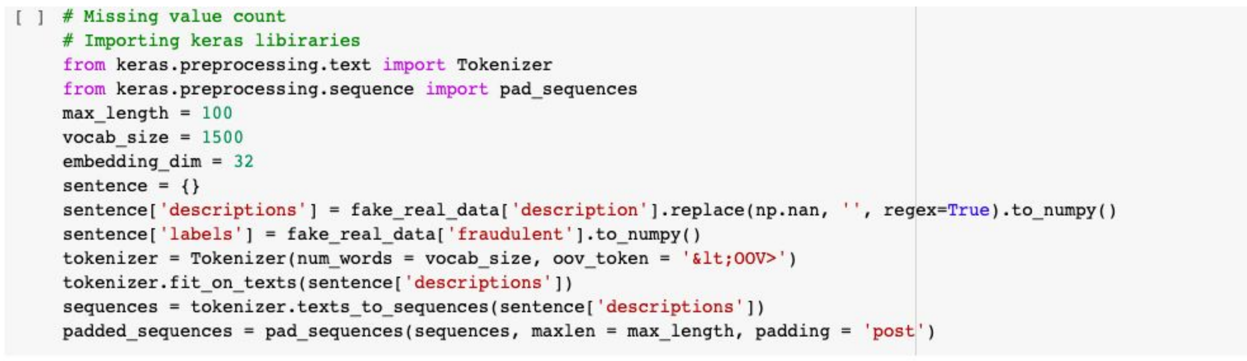
- Masked Language Modeling
- Next Sentence Prediction

Let’s understand both of these tasks in a little more detail!

Data Visualization: In my initial approach we tried to clean the data as much as possible to retrieve most useful data for my model to work properly. Below are the figures that depict the visualization and cleaning carried out to understand the entire dataset properly. Below graphs signifies how different attributes are dependent on each other which can help us derive useful insights.

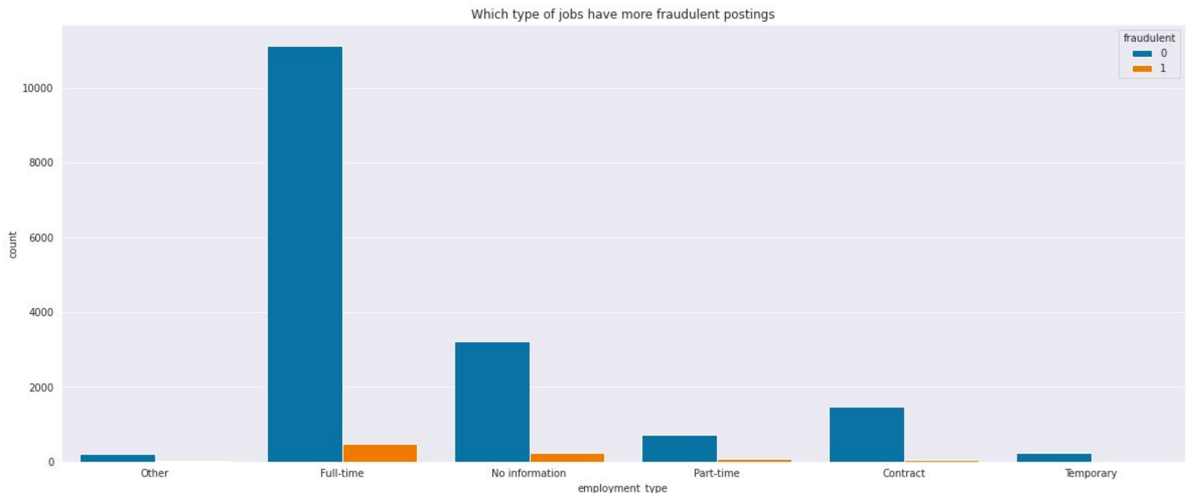


Visualisation of the missing value in the entire dataset

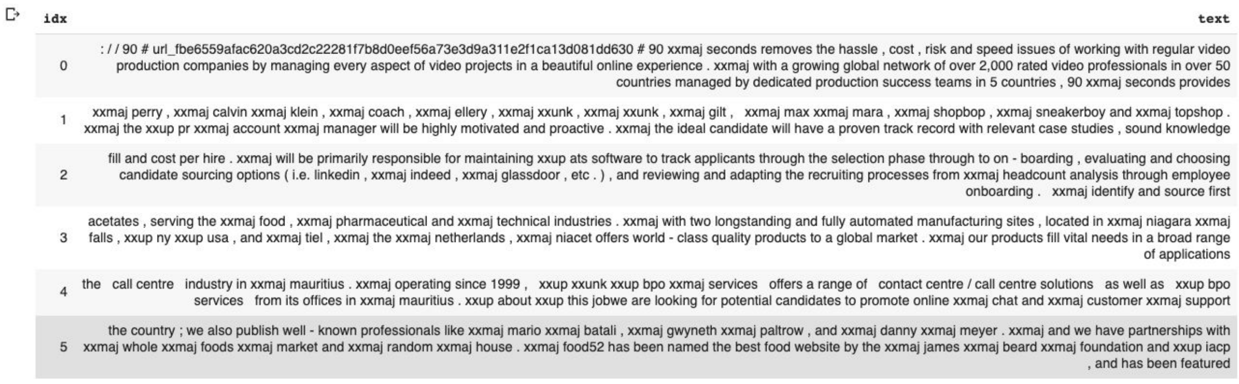


```
# Missing value count
# Importing keras libiraries
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
max_length = 100
vocab_size = 1500
embedding_dim = 32
sentence = {}
sentence['descriptions'] = fake_real_data['description'].replace(np.nan, '', regex=True).to_numpy()
sentence['labels'] = fake_real_data['fraudulent'].to_numpy()
tokenizer = Tokenizer(num_words = vocab_size, oov_token = '&lt;OOV>')
tokenizer.fit_on_texts(sentence['descriptions'])
sequences = tokenizer.texts_to_sequences(sentence['descriptions'])
padded_sequences = pad_sequences(sequences, maxlen = max_length, padding = 'post')
```

Missing value Count



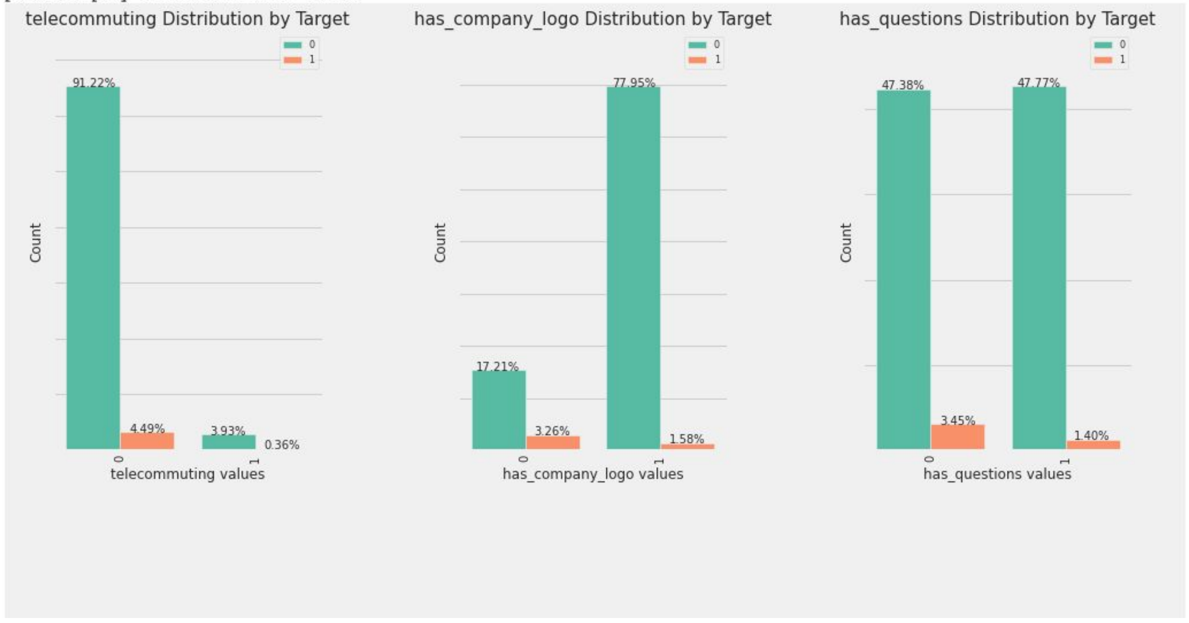
Employment type vs count visualization

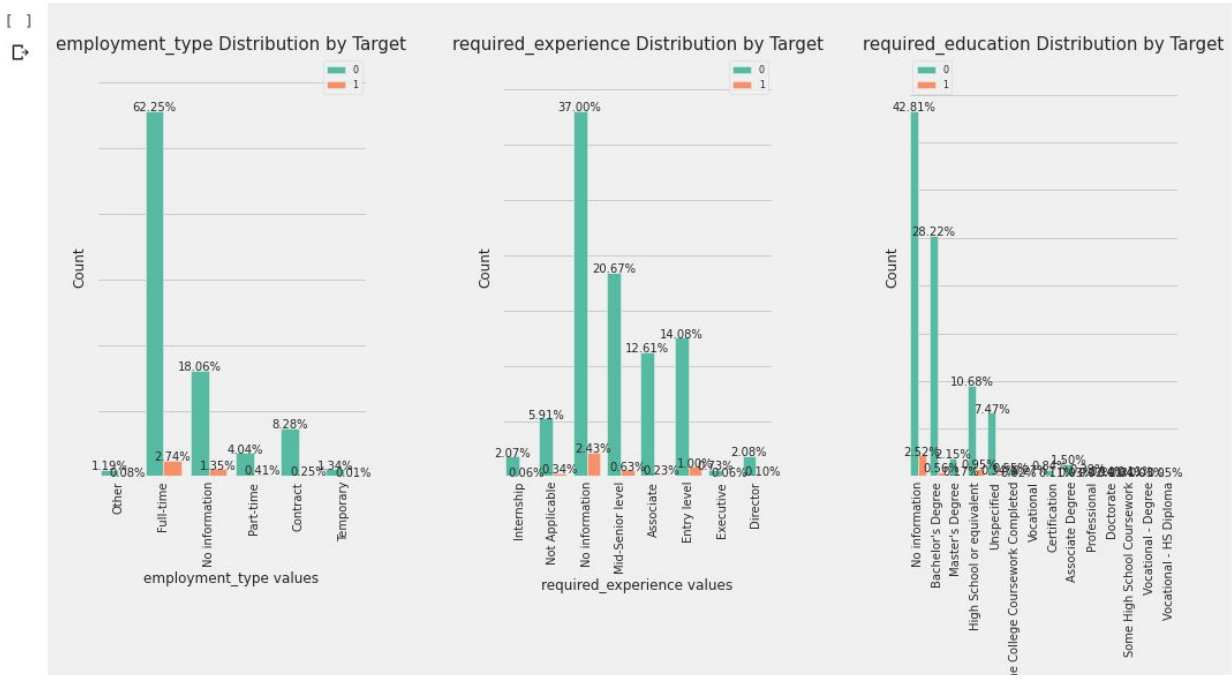


idx	text
0	: / / 90 # url_fbe6559afac620a3cd2c22281f7b8d0eef56a73e3d9a311e2f1ca13d081dd630 # 90 xxmaj seconds removes the hassle , cost , risk and speed issues of working with regular video production companies by managing every aspect of video projects in a beautiful online experience . xxmaj with a growing global network of over 2,000 rated video professionals in over 50 countries managed by dedicated production success teams in 5 countries , 90 xxmaj seconds provides
1	xxmaj perry , xxmaj calvin xxmaj klein , xxmaj coach , xxmaj ellery , xxmaj xxunk , xxmaj xxunk , xxmaj gilt , xxmaj max xxmaj mara , xxmaj shopbop , xxmaj sneakerboy and xxmaj topshop . xxmaj the xxup pr xxmaj account xxmaj manager will be highly motivated and proactive . xxmaj the ideal candidate will have a proven track record with relevant case studies , sound knowledge
2	fill and cost per hire . xxmaj will be primarily responsible for maintaining xxup ats software to track applicants through the selection phase through to on - boarding , evaluating and choosing candidate sourcing options (i.e. linkedin , xxmaj indeed , xxmaj glassdoor , etc .) , and reviewing and adapting the recruiting processes from xxmaj headcount analysis through employee onboarding . xxmaj identify and source first
3	acetates , serving the xxmaj food , xxmaj pharmaceutical and xxmaj technical industries . xxmaj with two longstanding and fully automated manufacturing sites , located in xxmaj niagara xxmaj falls , xxup ny xxup usa , and xxmaj tiel , xxmaj the xxmaj netherlands , xxmaj niacet offers world - class quality products to a global market . xxmaj our products fill vital needs in a broad range of applications
4	the call centre industry in xxmaj mauritius . xxmaj operating since 1999 , xxup xxunk xxup bpo xxmaj services offers a range of contact centre / call centre solutions as well as xxup bpo services from its offices in xxmaj mauritius . xxup about xxup this jobwe are looking for potential candidates to promote online xxmaj chat and xxmaj customer xxmaj support
5	the country ; we also publish well - known professionals like xxmaj mario xxmaj batali , xxmaj gwyneth xxmaj paltrow , and xxmaj danny xxmaj meyer . xxmaj and we have partnerships with xxmaj whole xxmaj foods xxmaj market and xxmaj random xxmaj house . xxmaj food52 has been named the best food website by the xxmaj james xxmaj beard xxmaj foundation and xxup iacp , and has been featured

Creating text list of few columns data

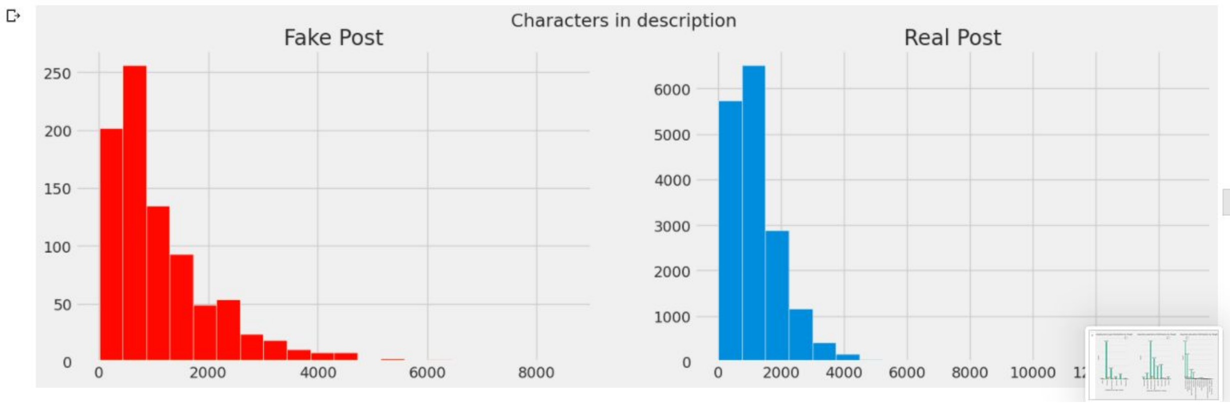
```
posx and posy should be finite values
```

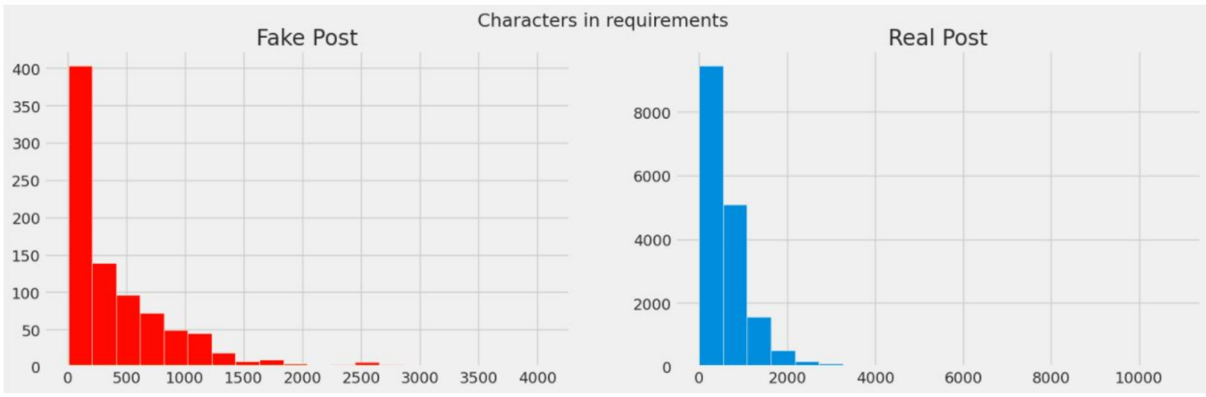


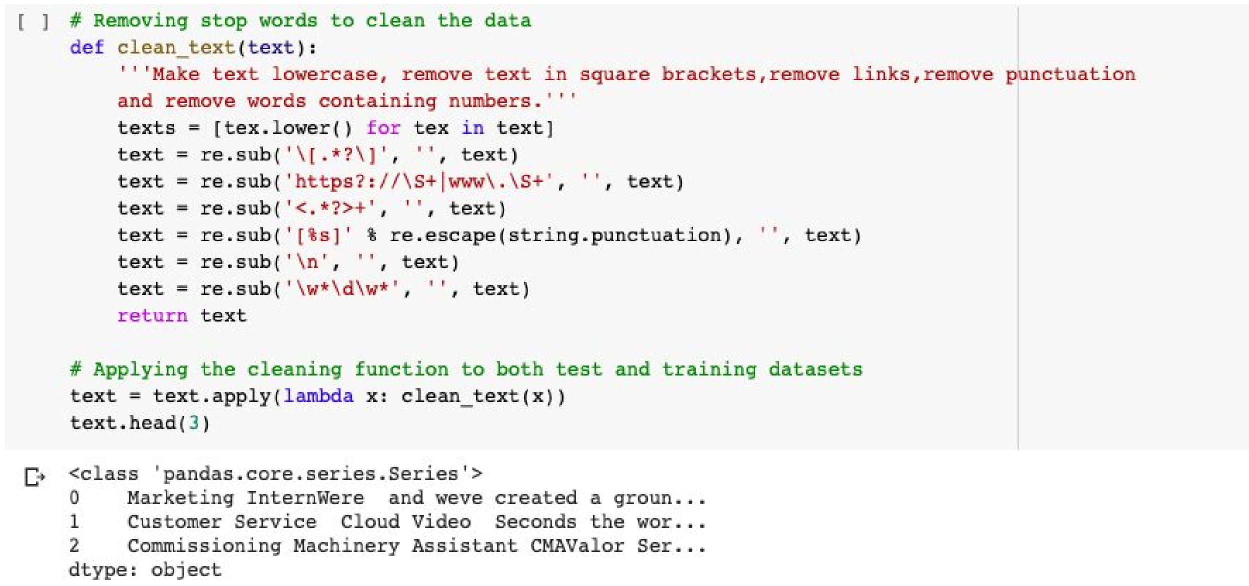


Visualizing categorical variable by target

Visualizing Posts with Characters with different column data







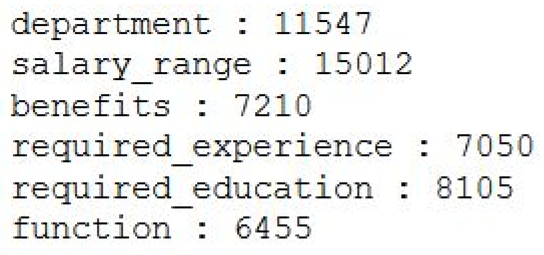
```
# Removing stop words to clean the data
def clean_text(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    texts = [tex.lower() for tex in text]
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

# Applying the cleaning function to both test and training datasets
text = text.apply(lambda x: clean_text(x))
text.head(3)
```

```
<class 'pandas.core.series.Series'>
0    Marketing InternWere  and weve created a groun...
1    Customer Service  Cloud Video  Seconds the wor...
2    Commissioning Machinery Assistant CMAValor Ser...
dtype: object
```

Removing stop words to clean the data

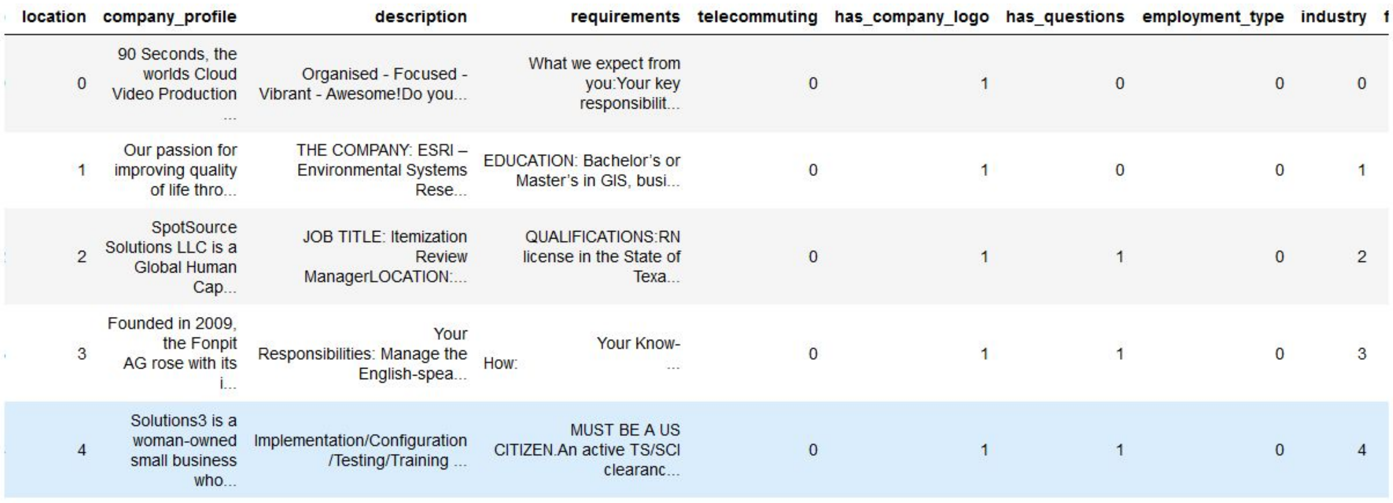
3.4) XGBoost [9-1]:

- XGBoost is a decision tree based model. It used a Gradient boosting framework for classification problems.
- XGBoost gives good performance when the data is well structured and small. As our dataset has approximately 18K records, XGBoost can be an appropriate model to implement
- Advantage of using models with predefined rules is that we can avoid overfitting the data specifically by using certain combinations of the rules.
- The data consists of 4 descriptive, untokenizable columns like Company_profile, Description, Benefits and requirements and non descriptive , tokenizable columns like job_id, Title, location, department, salary_range, telecommuting, has_company_logo, has_questions, employment_type, required_experience, required_education, industry, function and fraudulent
- Two Approaches were used for the implementation of the XGBoost,
- **Model 1 :**
    - Data was preprocess to remove Null values from the attributes
    - Attributes with at most 30% Null values was considered for the dataset Following attributes with more than 30% overall null values were removed



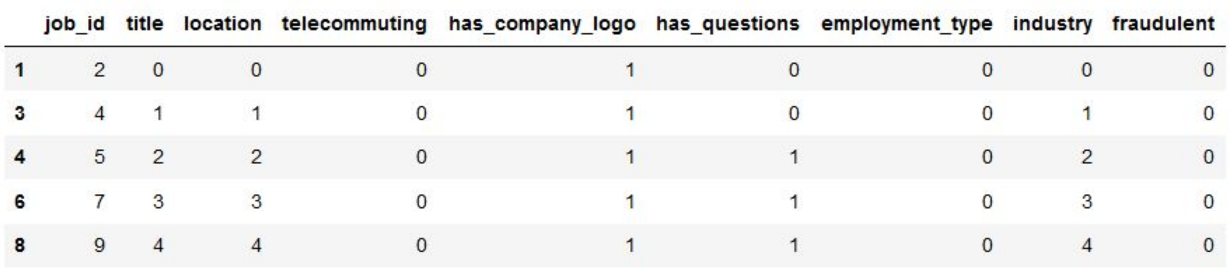
```
department : 11547
salary_range : 15012
benefits : 7210
required_experience : 7050
required_education : 8105
function : 6455
```

    - Non numerical attributes like title, location, employment_type and industry were encoded to produce numerical attributes for better mapping of weights to classes



location	company_profile	description	requirements	telecommuting	has_company_logo	has_questions	employment_type	industry
0	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we expect from you:Your key responsibilit...	0	1	0	0	0
1	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Bachelor's or Master's in GIS, busi...	0	1	0	0	1
2	SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATIONS:RN license in the State of Texa...	0	1	1	0	2
3	Founded in 2009, the Fonpit AG rose with its i...	Your Responsibilities: Manage the English-spea...	Your Know-How: ...	0	1	1	0	3
4	Solutions3 is a woman-owned small business who...	Implementation/Configuration/Testing/Training ...	MUST BE A US CITIZEN.An active TS/SCI clearanc...	0	1	1	0	4

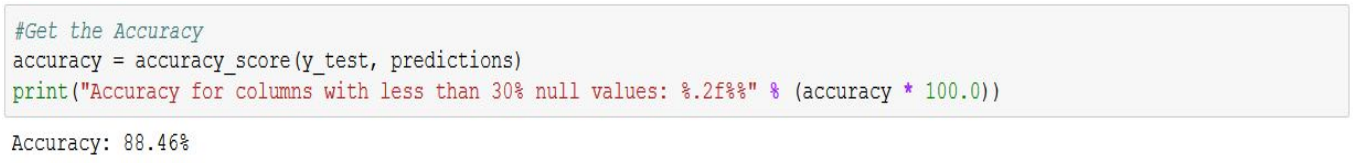
- Descriptive columns like company_profile, description and requirements were dropped from the dataset for training



	job_id	title	location	telecommuting	has_company_logo	has_questions	employment_type	industry	fraudulent
1	2	0	0	0	1	0	0	0	0
3	4	1	1	0	1	0	0	1	0
4	5	2	2	0	1	1	0	2	0
6	7	3	3	0	1	1	0	3	0
8	9	4	4	0	1	1	0	4	0

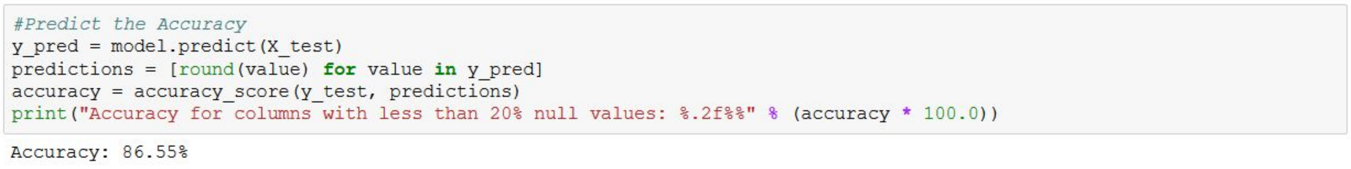
- job_id, title, location, telecommuting, has_company_logo, has_questions, employment_type and industry were considered for training_data set and fraudulent was used for prediction. Seed was taken as 7 and training and testing data was split to 80 - 20 ratio.
- XGBoost classifier was used to fit that training data to fraudulent training set
- Fitted model was used to predict fraudulent data using test data
- **Results :**

For at most 30% Null valued attributes records :
Accuracy : 88.46%



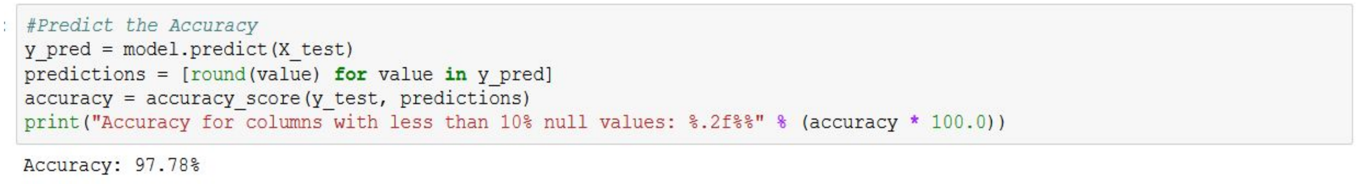
```
#Get the Accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy for columns with less than 30% null values: %.2f%%" % (accuracy * 100.0))

Accuracy: 88.46%
```

For at most 20% Null valued attributes records :
Accuracy : 86.55%



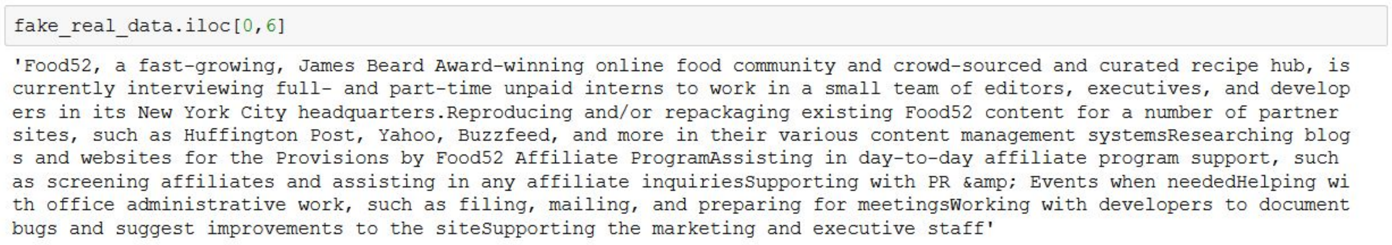
```
#Predict the Accuracy
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
print("Accuracy for columns with less than 20% null values: %.2f%%" % (accuracy * 100.0))

Accuracy: 86.55%
```

For at most 10% Null valued attributes records :
Accuracy : 97.78%



```
#Predict the Accuracy
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
print("Accuracy for columns with less than 10% null values: %.2f%%" % (accuracy * 100.0))

Accuracy: 97.78%
```

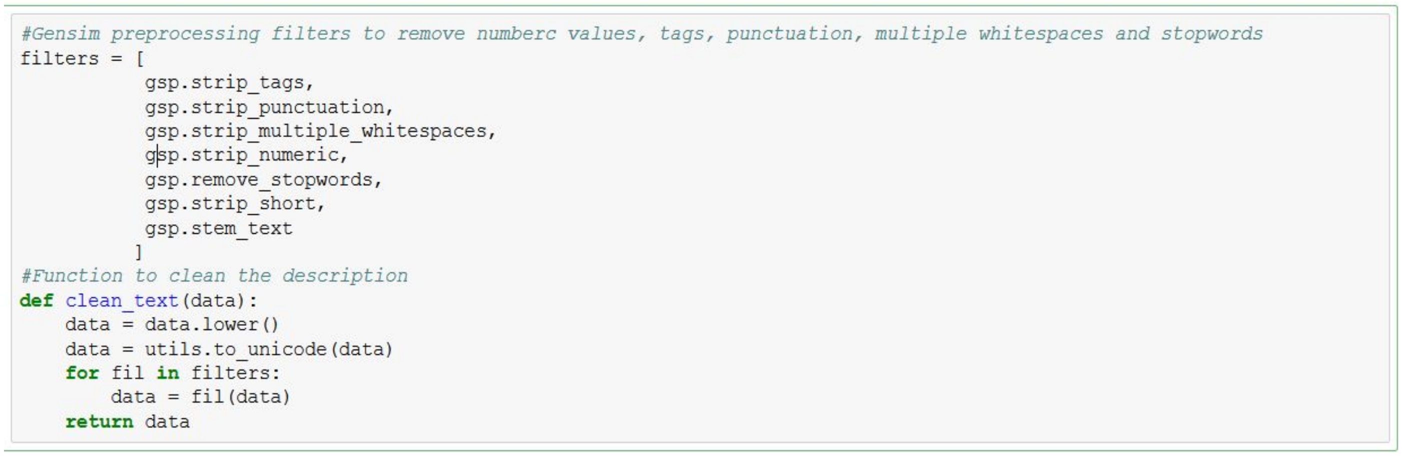
- **Model 2 :**
  - Data attributes were preprocessed and Null valued records were dropped
  - Only descriptive data were considered for training. For this case, Job_description and Compnay_profile
  - Example of Job_description



```
fake_real_data.iloc[0,6]
```

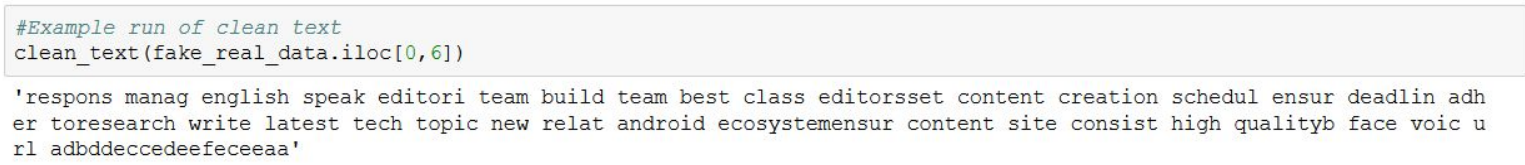
```
'Food52, a fast-growing, James Beard Award-winning online food community and crowd-sourced and curated recipe hub, is
currently interviewing full- and part-time unpaid interns to work in a small team of editors, executives, and develop
ers in its New York City headquarters.Reproducing and/or repackaging existing Food52 content for a number of partner
sites, such as Huffington Post, Yahoo, Buzzfeed, and more in their various content management systemsResearching blog
s and websites for the Provisions by Food52 Affiliate ProgramAssisting in day-to-day affiliate program support, such
as screening affiliates and assisting in any affiliate inquiriesSupporting with PR &amp; Events when neededHelping wi
th office administrative work, such as filing, mailing, and preparing for meetingsWorking with developers to document
bugs and suggest improvements to the siteSupporting the marketing and executive staff'
```

  - Descriptive data is filtered to remove stop words, tags, Punctuations, numerics, Multiple whitespaces, stem texts using Gensim library.



```
#Gensim preprocessing filters to remove numberc values, tags, punctuation, multiple whitespaces and stopwords
filters = [
           gsp.strip_tags,
           gsp.strip_punctuation,
           gsp.strip_multiple_whitespaces,
           gsp.strip_numeric,
           gsp.remove_stopwords,
           gsp.strip_short,
           gsp.stem_text
          ]
#Function to clean the description
def clean_text(data):
    data = data.lower()
    data = utils.to_unicode(data)
    for fil in filters:
        data = fil(data)
    return data
```

  - Below is an example of filtering on previously shown description. We can change the filters into different combinations for different representations of attribute records.

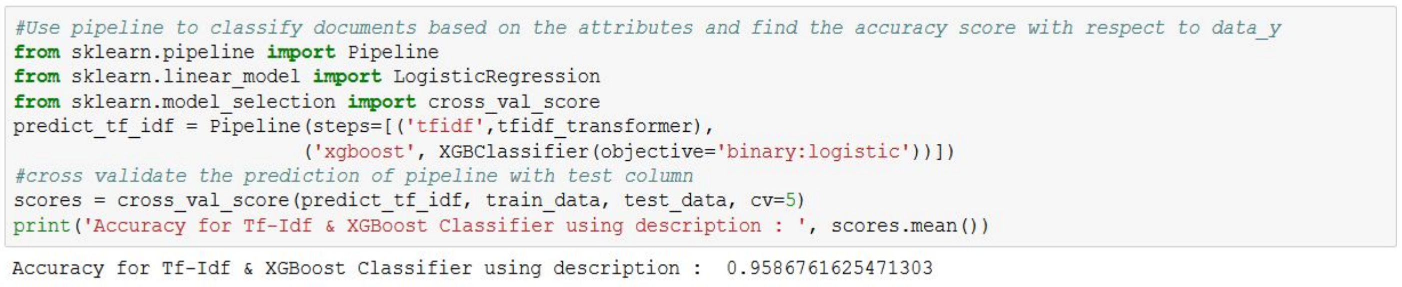


```
#Example run of clean text
clean_text(fake_real_data.iloc[0,6])
```

```
'respons manag english speak editori team build team best class editorsset content creation schedul ensur deadlin adh
er toresearch write latest tech topic new relat android ecosystemensur content site consist high qualityb face voic u
rl adbddeccedeefeceeaa'
```

  - TF-IDF vectorization is used to fit the training samples and transform them. Training sample is the filtered code and the testing sample is the fraudulent column.
  - Pipelining is done using the TF-IDF fitted transformed training set and XGBoost classifier model to predict the Fraudulent score.

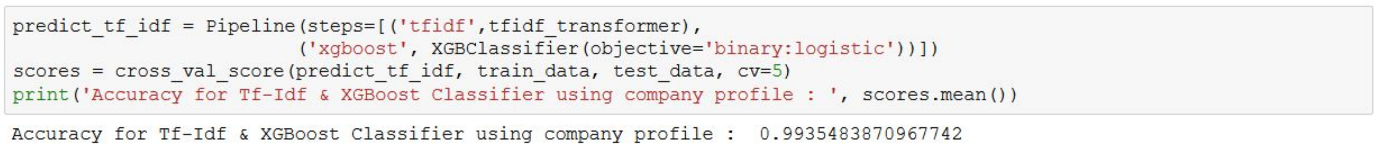
- Predicted data is validated against the entire fraudulent testing data for results.
- **Results :**
  For Description attribute : Accuracy = 95.86%



```
#Use pipeline to classify documents based on the attributes and find the accuracy score with respect to data_y
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
predict_tf_idf = Pipeline(steps=[('tfidf',tfidf_transformer),
                         ('xgboost', XGBClassifier(objective='binary:logistic'))])
#cross validate the prediction of pipeline with test column
scores = cross_val_score(predict_tf_idf, train_data, test_data, cv=5)
print('Accuracy for Tf-Idf & XGBoost Classifier using description : ', scores.mean())
```

```
Accuracy for Tf-Idf & XGBoost Classifier using description :  0.9586761625471303
```

For Company_profile attribute : Accuracy = 99.35%

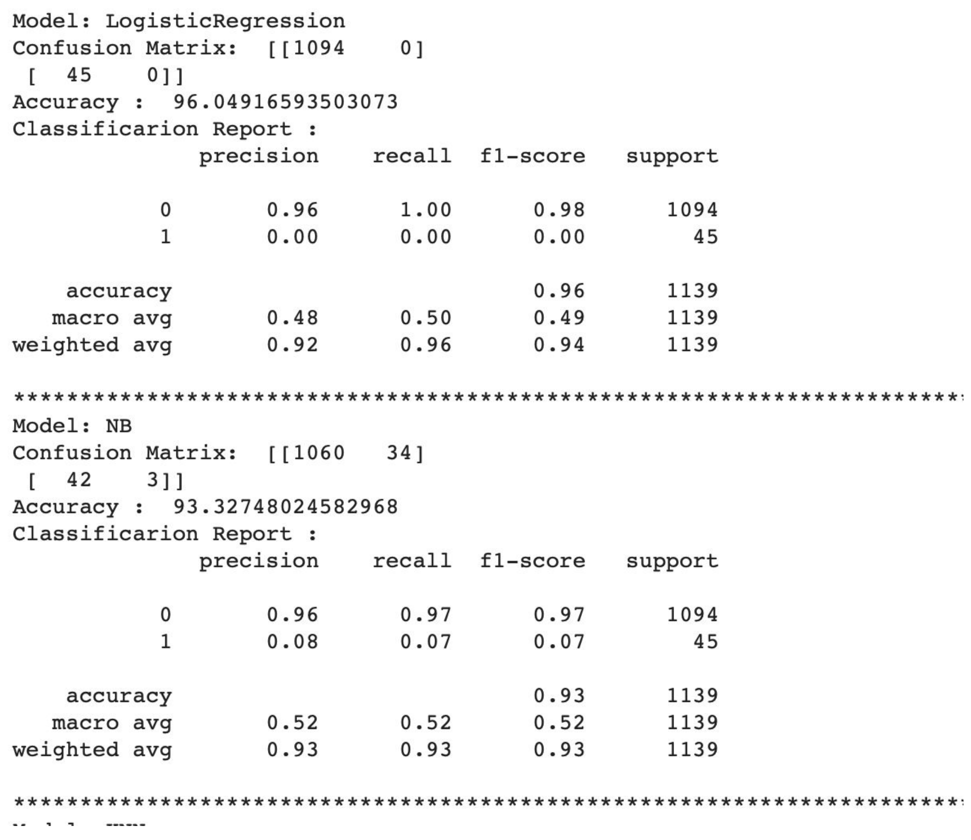


```
predict_tf_idf = Pipeline(steps=[('tfidf',tfidf_transformer),
                         ('xgboost', XGBClassifier(objective='binary:logistic'))])
scores = cross_val_score(predict_tf_idf, train_data, test_data, cv=5)
print('Accuracy for Tf-Idf & XGBoost Classifier using company profile : ', scores.mean())
```

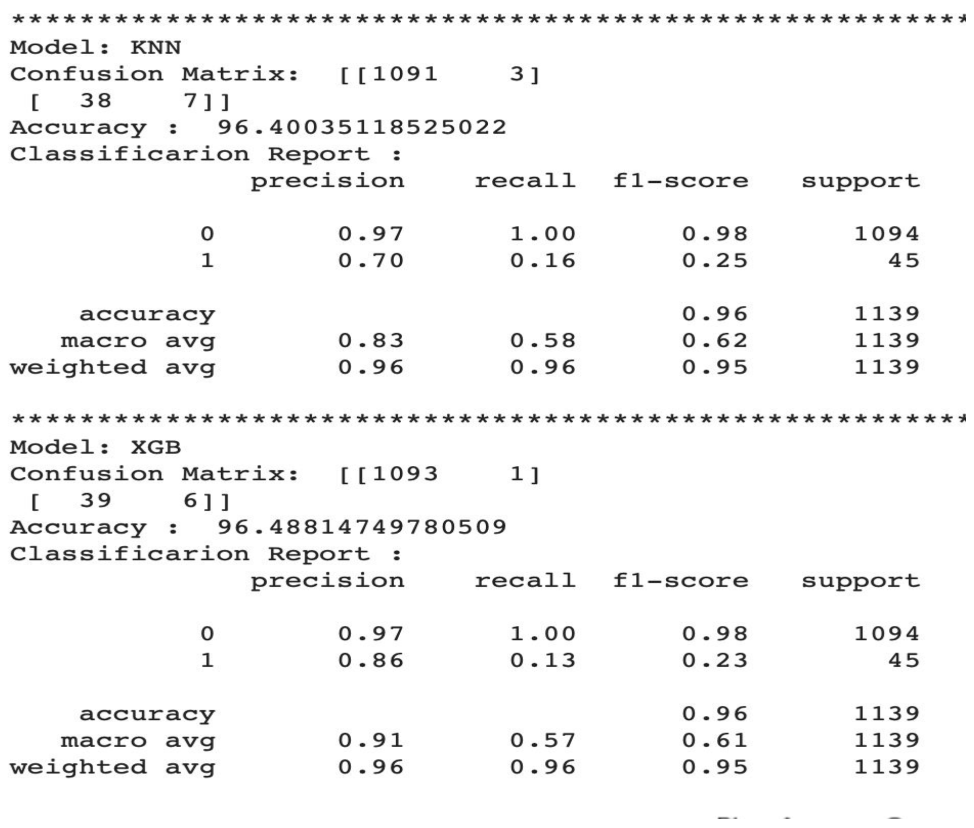
```
Accuracy for Tf-Idf & XGBoost Classifier using company profile :  0.9935483870967742
```

4. Preliminary results

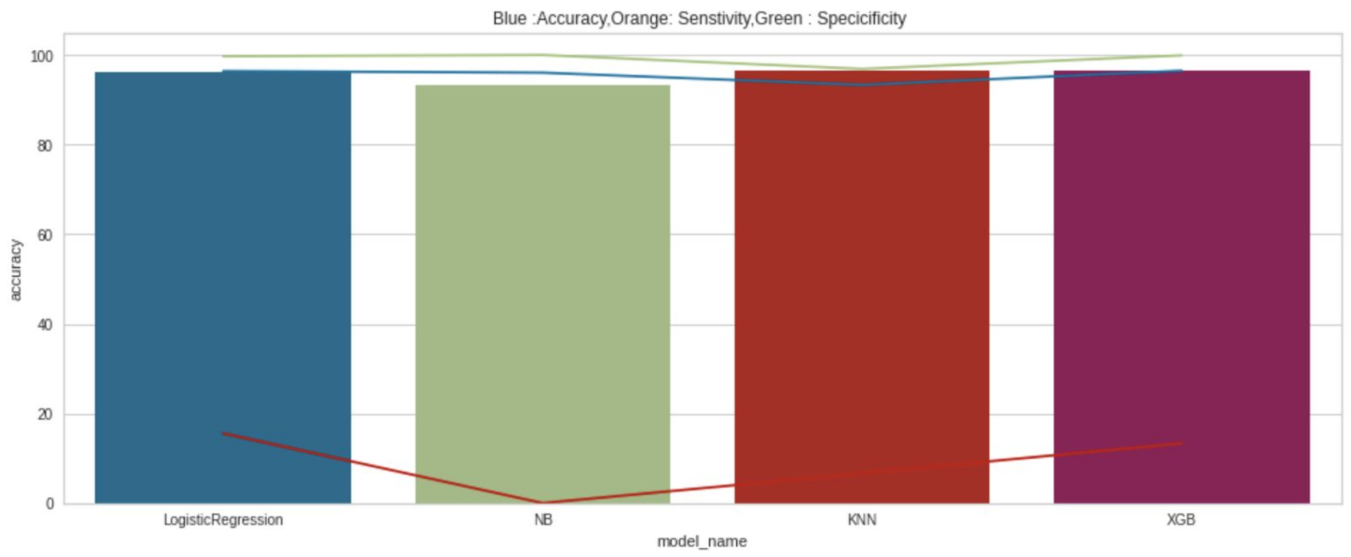
For preliminary analysis we wanted to see how different models (not necessarily great for text analysis) would perform on this kind of data. Thus, we first label encoded all of the columns, and tested models like Logistic Regression, K Nearest Neighbors and GaussianNB on the data.
An important thing to note for this is we did not do much data cleaning and preprocessing for this analysis. We did not even balance the dataset (which is extremely important for such datasets with high imbalance). The accuracy seems to be pretty high for these models, but we suspect that is because of the imbalance of the dataset. We aim to achieve a comparable or higher accuracy after balancing the data with the models we'll be working on.
Following are the preliminary results we obtained for the above models. (These models are not part of the models we'll be working on).



```
Model: LogisticRegression
Confusion Matrix:  [[1094    0]
 [  45    0]]
Accuracy :  96.04916593503073
Classificarion Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      1094
           1       0.00      0.00      0.00        45

    accuracy                           0.96      1139
   macro avg       0.48      0.50      0.49      1139
weighted avg       0.92      0.96      0.94      1139

************************************************************************
Model: NB
Confusion Matrix:  [[1060   34]
 [  42    3]]
Accuracy :  93.32748024582968
Classificarion Report :
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      1094
           1       0.08      0.07      0.07        45

    accuracy                           0.93      1139
   macro avg       0.52      0.52      0.52      1139
weighted avg       0.93      0.93      0.93      1139

************************************************************************
```



```
*********************************************************
Model: KNN
Confusion Matrix:  [[1091    3]
 [  38    7]]
Accuracy :  96.40035118525022
Classificarion Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1094
           1       0.70      0.16      0.25        45

    accuracy                           0.96      1139
   macro avg       0.83      0.58      0.62      1139
weighted avg       0.96      0.96      0.95      1139

*********************************************************
Model: XGB
Confusion Matrix:  [[1093    1]
 [  39    6]]
Accuracy :  96.48814749780509
Classificarion Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1094
           1       0.86      0.13      0.23        45

    accuracy                           0.96      1139
   macro avg       0.91      0.57      0.61      1139
weighted avg       0.96      0.96      0.95      1139
```



5. Experiment / Analysis

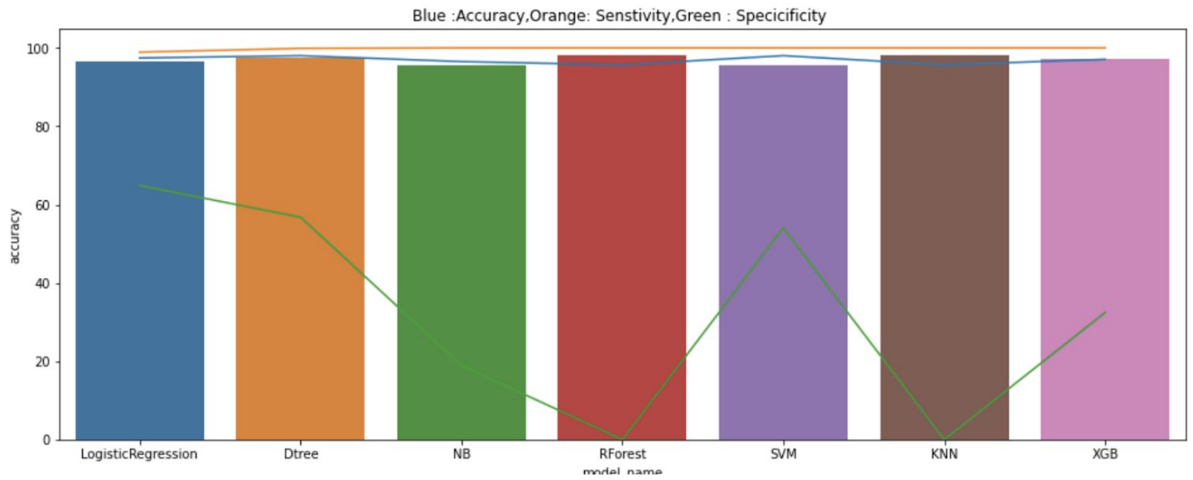
Testing accuracy

Logistic Regression	NB	KNN	XGB (Non descriptive)
96%	93%	96%	95.86%

Random Forest Classifier	TF-IDF	NLP ulmfit	XGB (Descriptive)
96%	98.32		97%

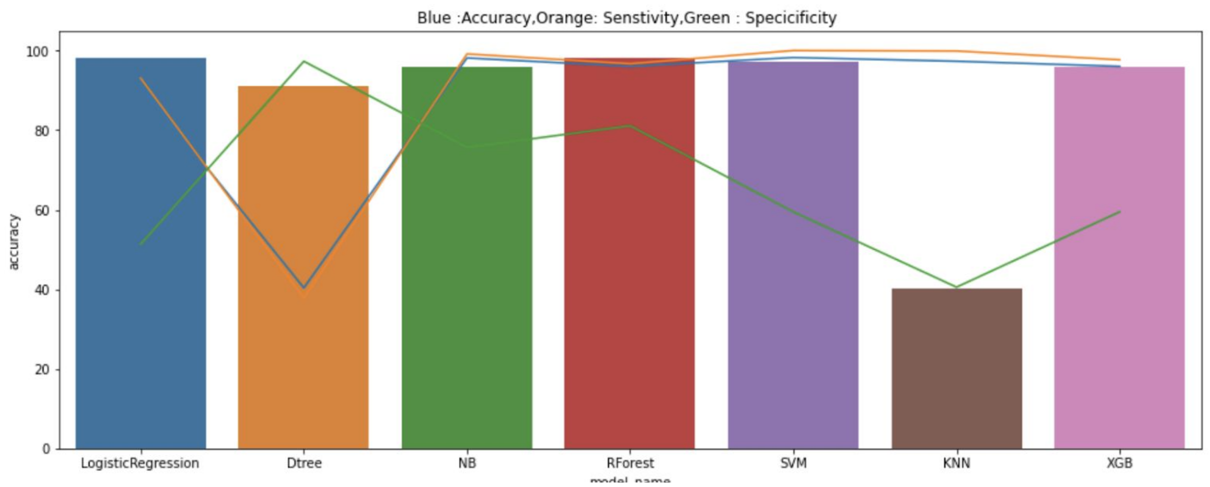
Performance of models based on TF-IDF vectorization on descriptive columns (before oversampling and undersampling)

Logistic regression	Decision Tree	Naive-Bayes	Random Forest	Support Vector Model (SVM)	KNN	XGBoost
96.49%	97.42%	95.67%	98.01%	95.67%	98.01%	97.07%



Performance of models based on TF-IDF vectorization on descriptive columns (after oversampling using SMOTE and undersampling using NearMiss)

Logistic regression	Decision Tree	Naive-Bayes	Random Forest	Support Vector Model (SVM)	KNN	XGBoost
98.12%	91.22%	96.02%	98.24%	97.31%	40.35%	96.02%



6. Comparisons

Performance comparision on XGBoost classifier vs Random Forest on Non descriptive columns

Accuracy for XGBoost = 97%

Accuracy for Random Forest = 96%

Observation for the XGBoost algorithm using TF-IDF pipelining with Gensim vs NLTK and TF-IDF vectorization using under sampling and oversampling is

Accuracy for XGBoost with TF-IDF Pipelining = 95.86

Accuracy for XGboost with TF-IDF Vectorization without sampling = 98.322

Accuracy for XGBoost with TF-IDF Vectorization with Undersampling majority class and oversampling minority class = 96.308

7. Conclusion

In this project we proposed multiple models to analyze the data and predict fake job posting from the data. Our model did a fairly good job at prediction with accuracy scores ranging from 93 - 99 %. We can use our model to predict and classify if a job posting is fake or real at this point. This can be extremely helpful for online job portals that host job postings from multiple sources in determining fake job posting with malicious intent. It also helps potential job applicants to secure their data from malicious use.

8. Repository

Following is the link to the Github repository for the CMPE 255 Data Mining project “Job posting analysis and fake vs real job posting detection”.

https://github.com/ShivanDesai/FakevsRealJobPostingDetection

The repository includes :

- Folder - Dataset : Dataset of Fake vs real job postings from The University of the Aegean
- Folder - Report : Milestone reports
- Folder - XGB-RandomForest : Performance comparison of XGBoost classifier and Random forest on factorized, non-descriptive, non-null data
- Folder - XGB-TFIDF : Performance comparison of XGBoost classifier using TF-IDF pipelining and Gensim and XG-Boost Classifier on TF-IDF Vectorized data using NLTK on Descriptive, non-null data
- DataPreprocessing_EDA.ipynb : Preprocessing of fake_real_data.csv, Analysis of columns, types of data present in columns, diversity of data, finding patterns in data using exploratory data analysis, identifying null values, removal of null values, Pearson's correlation on non null data, Pearson's Chi Square test on non null data
- Random_Forest_Classifier.ipynb : Implementation of the Random forest classifier model on the non descriptive attributes to classify the postings with testing accuracy of 96%.

- XGBoostV2.ipynb : Comparison of performance of XGBoost classification on non descriptive, non - null data with atmost 10%, 20% or 30% null valued attributes.
- XGBoostV3.ipynb : Comparison of performance of XGBoost classification on descriptive, non - null data with TF-IDF pipeline and Gensim NLP package on company_profile and job description attributes.

9. References

9.1 T Chen, C Guestrin
XGBoost: A Scalable Tree Boosting System
22nd ACM SIGKDD Int. Conf. 2016, 785.

9.2 Savan Patel
Machine Learning 101 - Random Forest Classifier

9.3 Leo Breiman and Adele Cutler
Random Forests, Salford Systems.