EE0005 Mini Project

Fraudulent Job Postings

Philip, Keng Soon, Hansel, Lee Hua

Contribution parts are listed at the end

Why this dataset?

Fake job postings do a lot of harm:

- Waste time
- Personal information lost

Our objectives:

- Identify fake job postings by sentiment analysis.
- Which model is the best? → metric scores, memory and speed
- Identify countries which are associated with fraud postings.
- Find associated features with a fraudulent and non-fraudulent posting.

NLP Pipeline

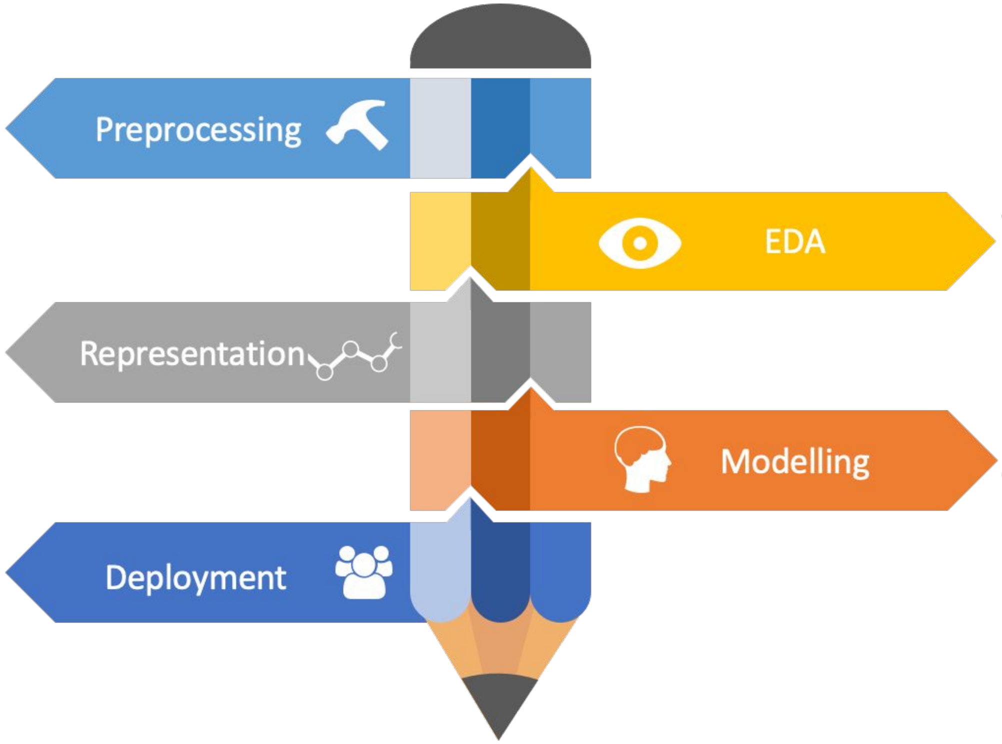
Text Preprocessing
Cleaning, Stemming, Contraction Removal, Special Char removal

EDA
Word2Vec Embedding enrichment, Misspell Removal, feature creation

Representation
Tokenization, text to sequence, padding sequences

Modelling
Models: Bi-LSTM/GRU, Attention, Capsule etc.

Deployment
Prediction and model evaluation



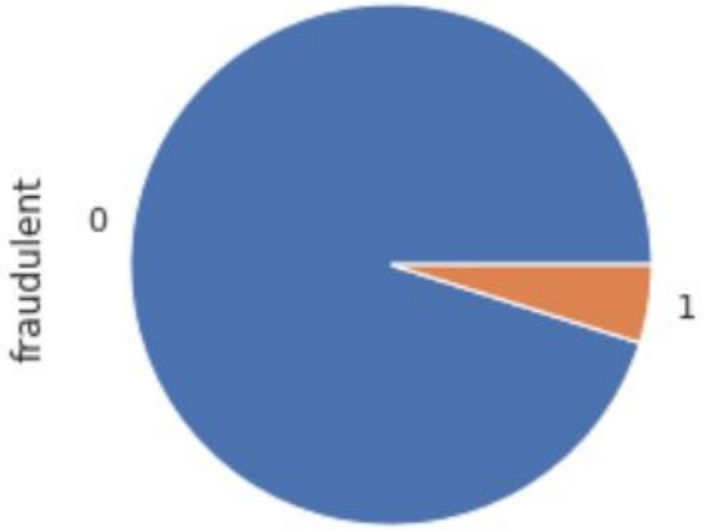
Exploratory Data Analysis

We will analyse the following with supporting visualization:

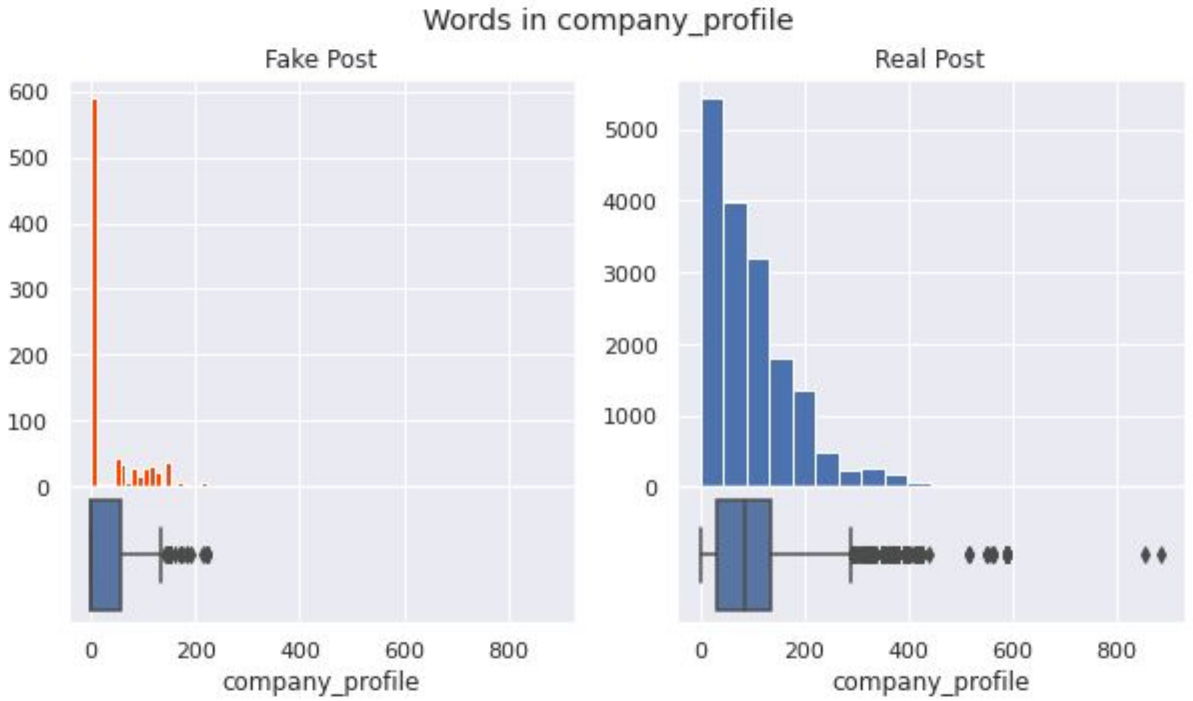
- Distributions
- Correlation
- Associated words with real / fraudulent posts
- Proportion of fraudulence from top 10 countries

Overview of dataset

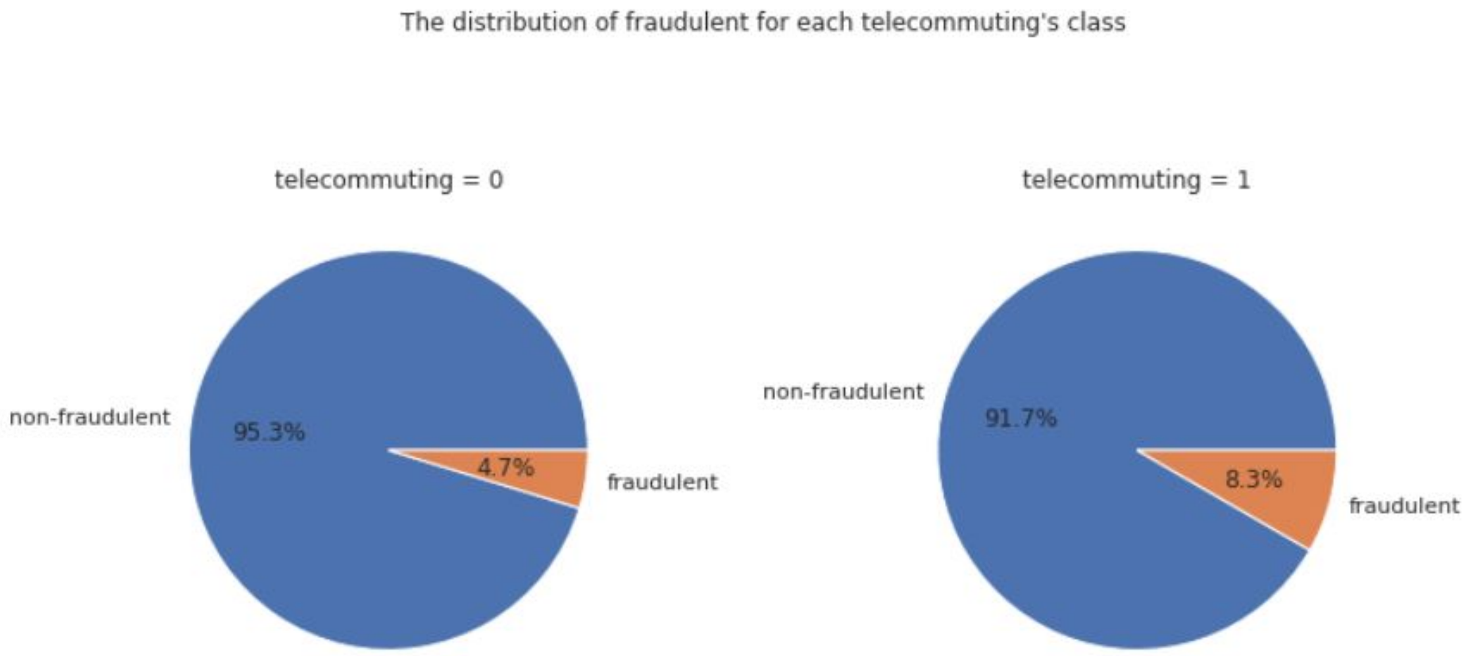
```
Percentage of fraudulent job postings = 4.84 %
```



Distributions of word count



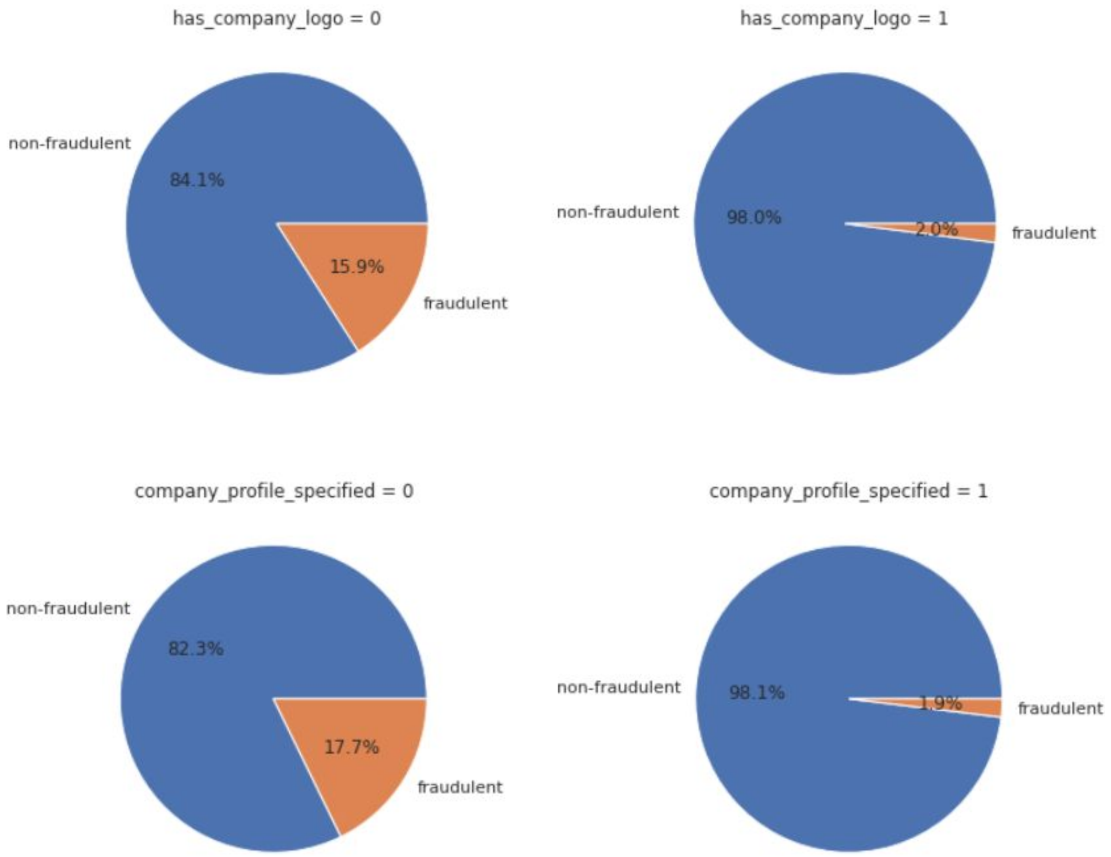
Distributions of binary features



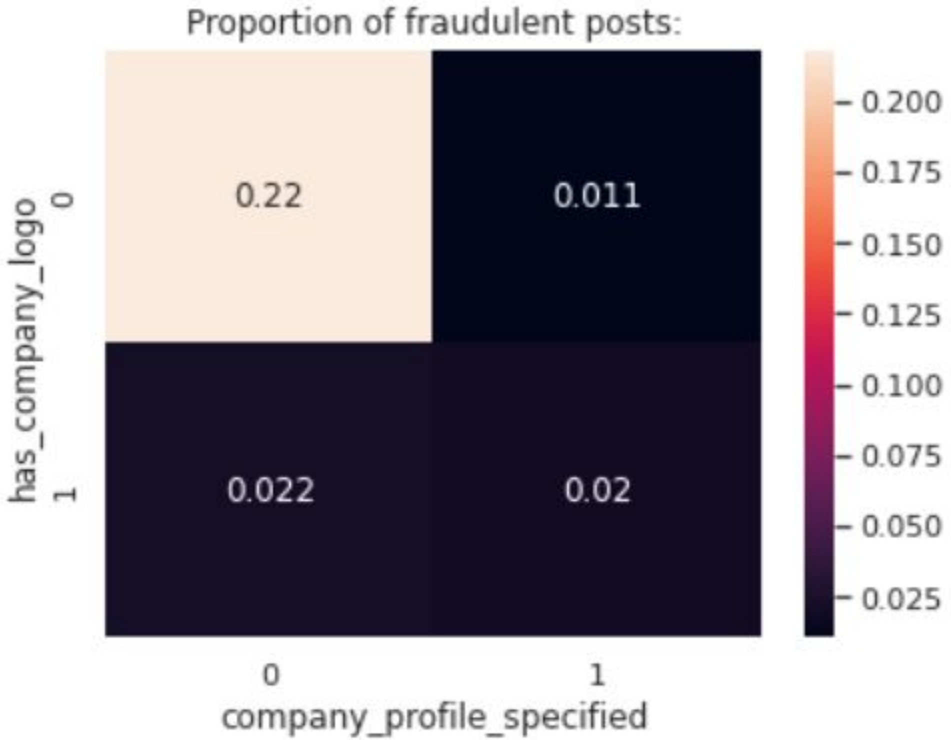
Hypothesis

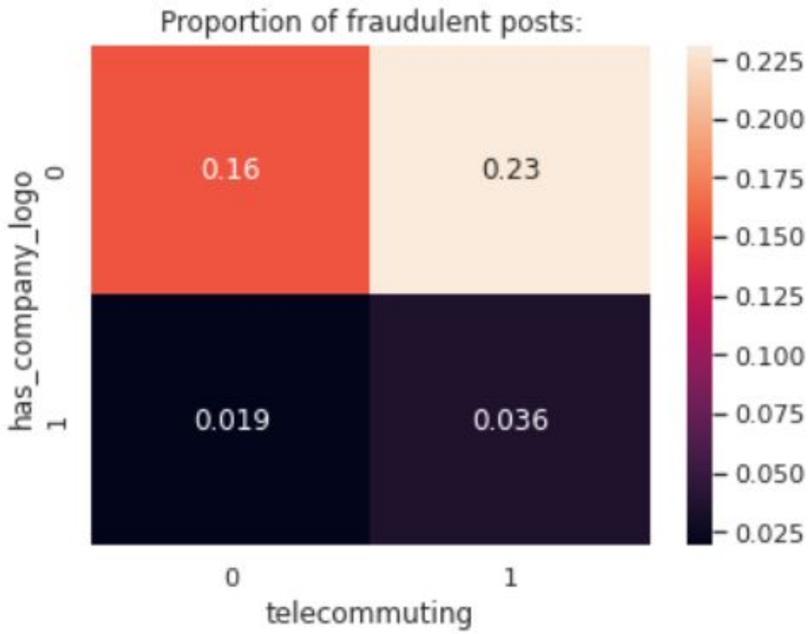
`has_company_logo`
`company_profile_specified`

Important features in fraud job postings

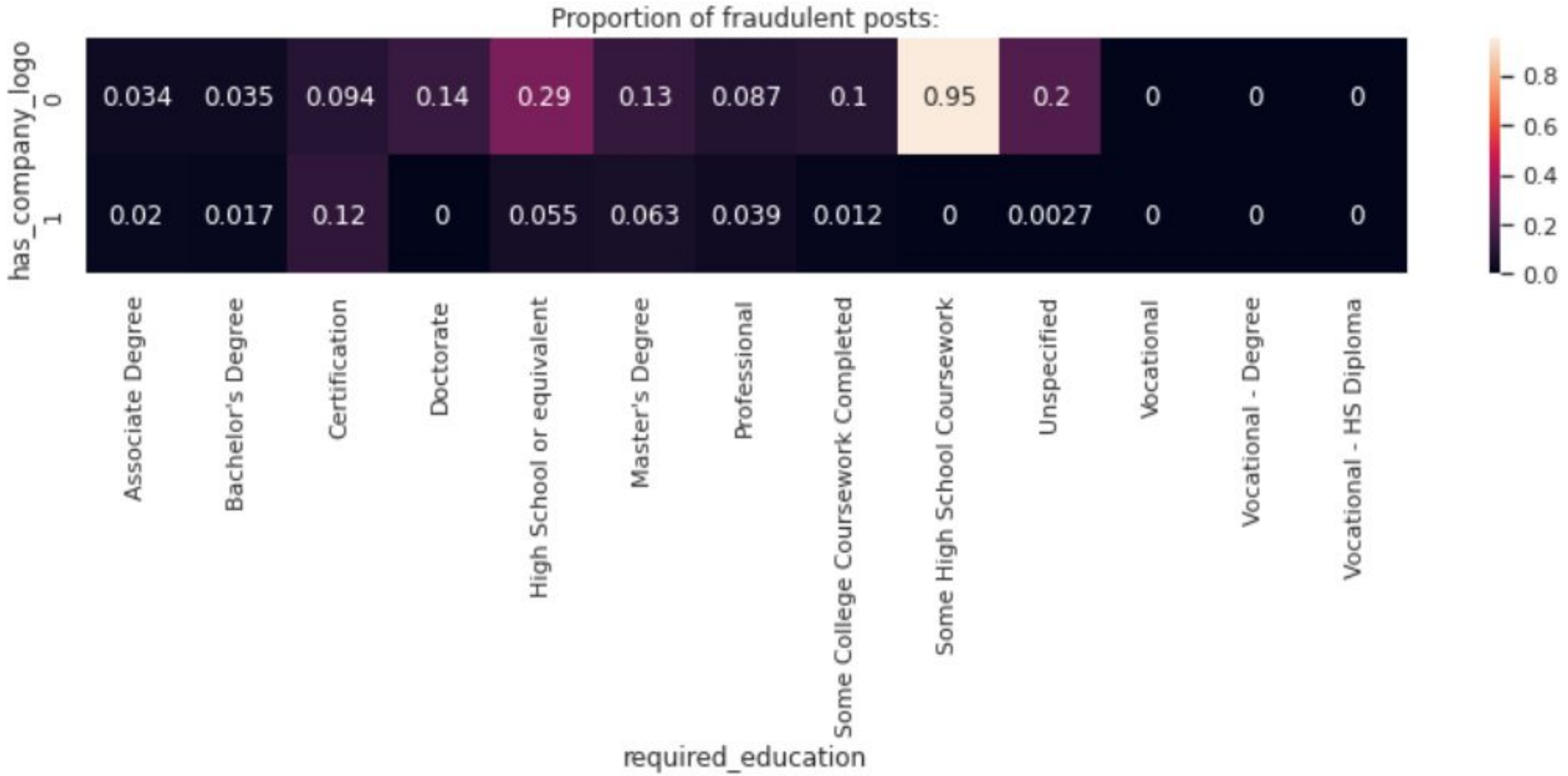


Correlation of has_company_logo (1)

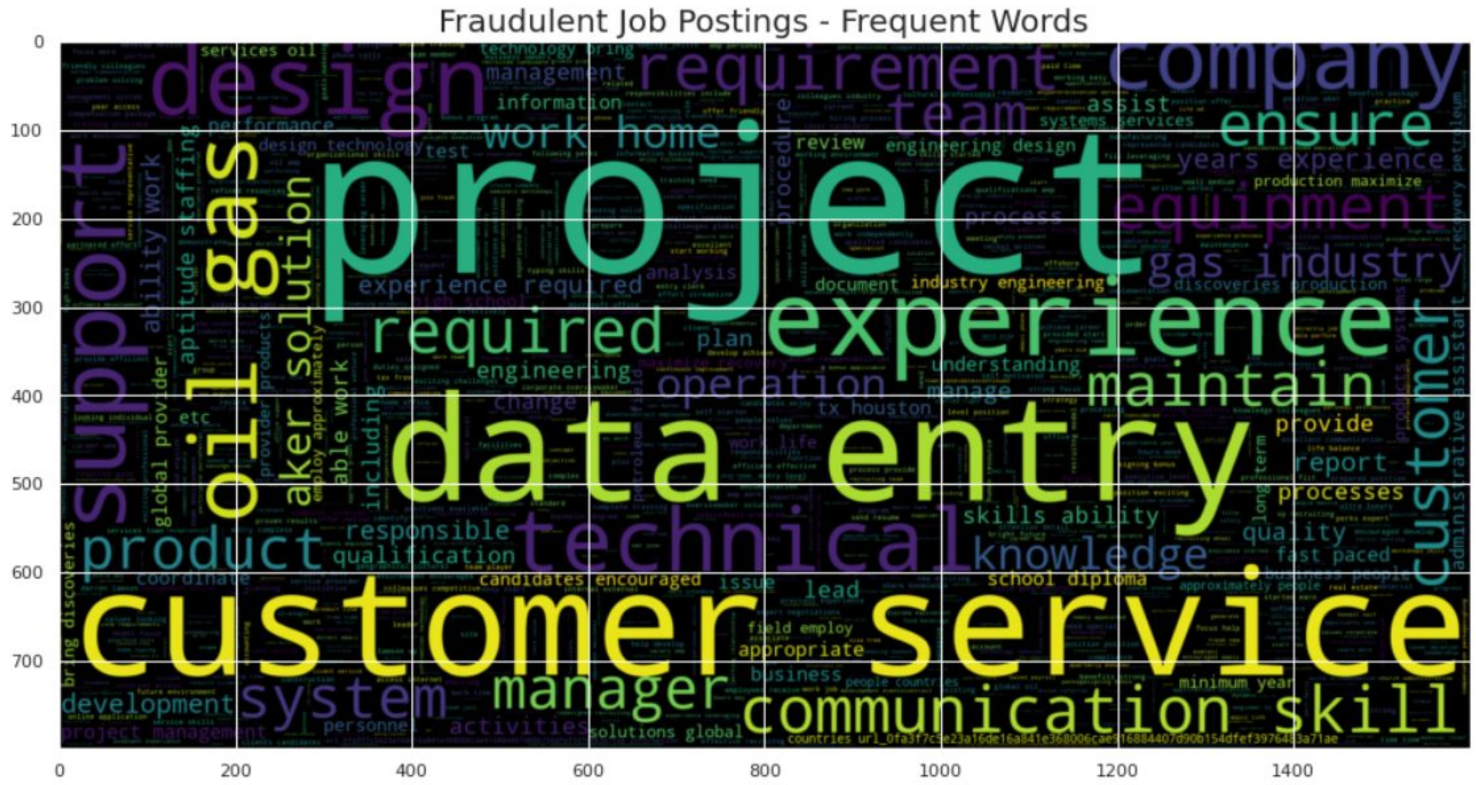




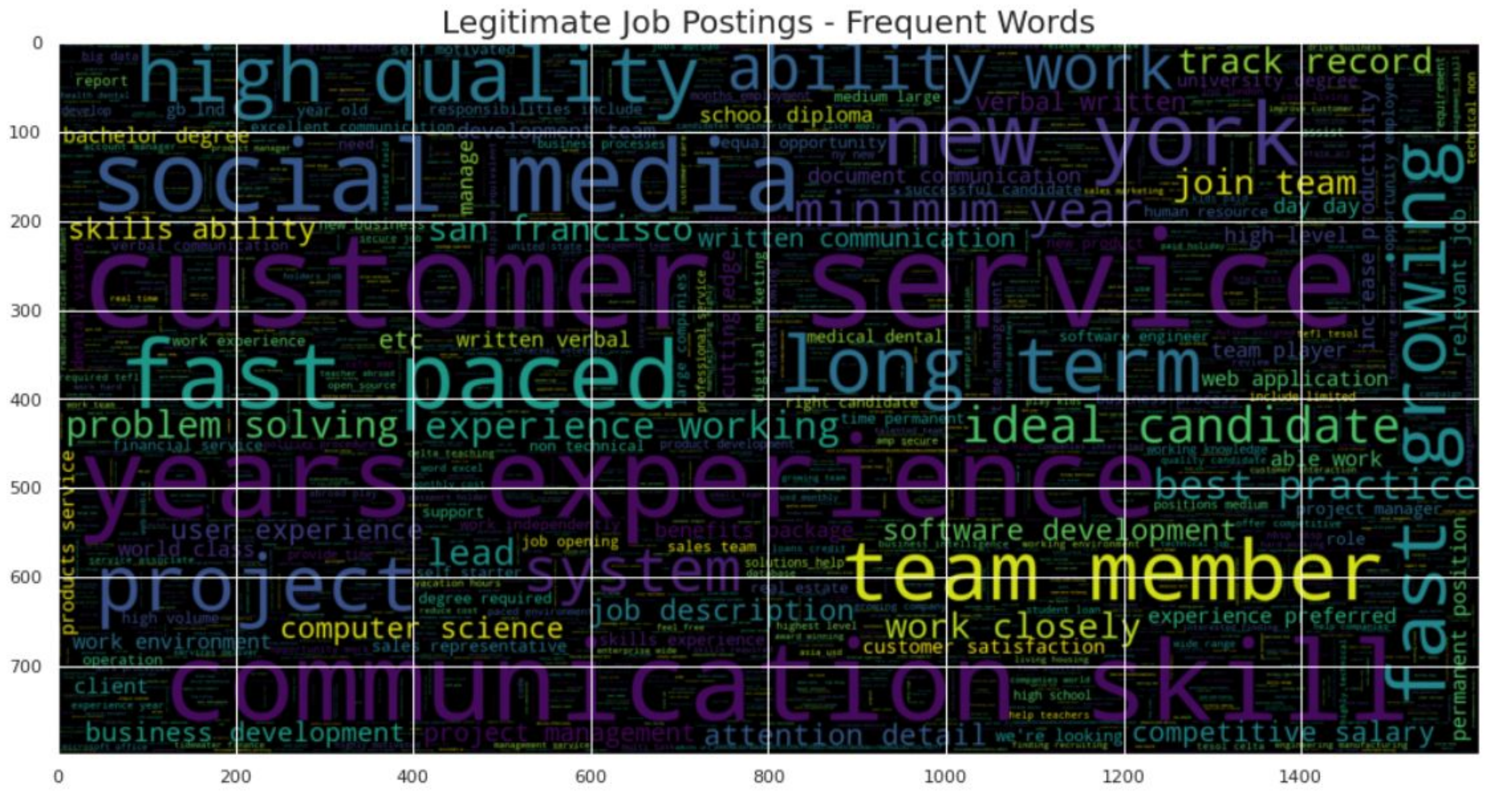
Correlation of has_company_logo (2)



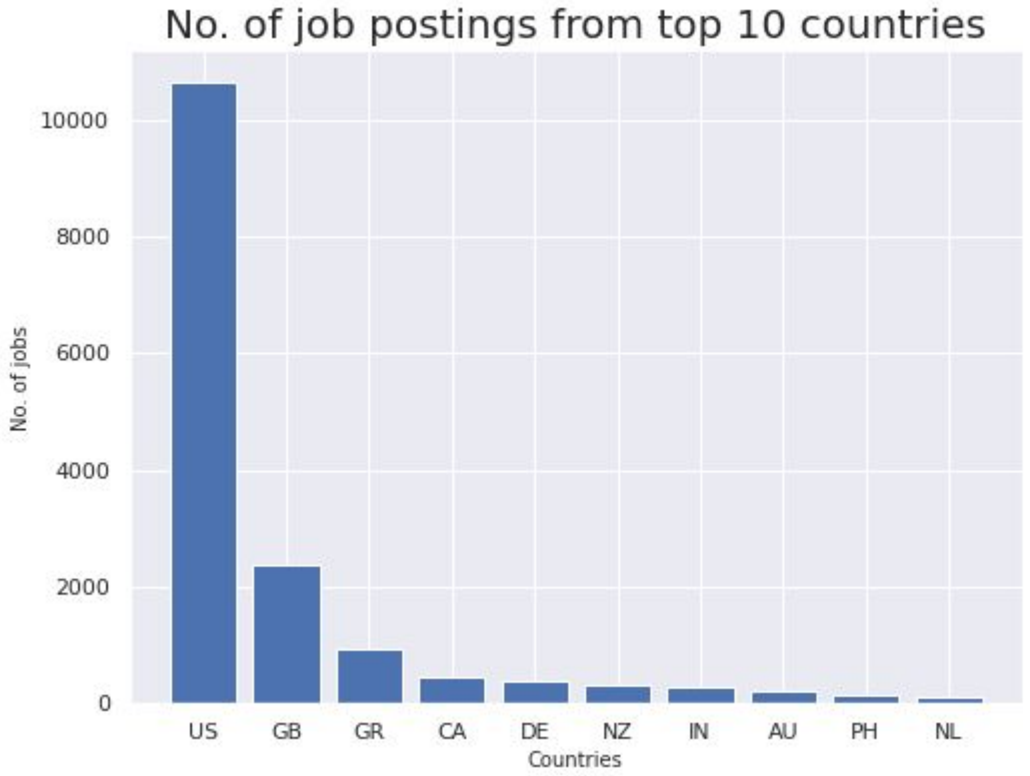
Associated words with fraudulent posts



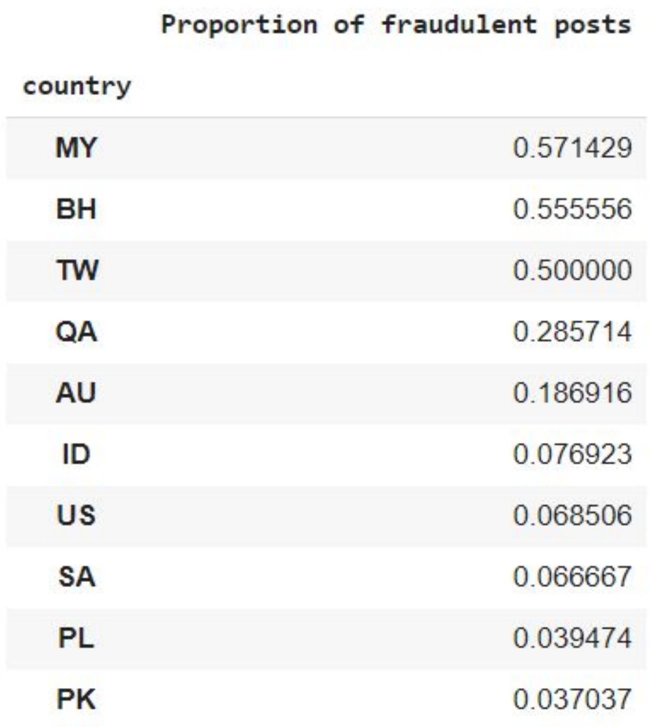
Associated words with real posts



Source of fraudulent job postings



Proportion of fraudulent job postings



country	Proportion of fraudulent posts
MY	0.571429
BH	0.555556
TW	0.500000
QA	0.285714
AU	0.186916
ID	0.076923
US	0.068506
SA	0.066667
PL	0.039474
PK	0.037037

Feature weights from model (in descending order):

```
['US', 'AU', 'PL', 'MY', 'PK', 'ID', 'BH', 'CA', 'ES', 'GB']
```

Difference in order due to random sampling of train-test data!

✓ *Identify countries which are associated with fraud postings*

Metrics & Oversampling

Dummy Classifier has good accuracy!

```
dummy_clf = DummyClassifier(strategy="most_frequent")
```

Accuracy: 0.9559 (from predicting all ‘0’)

- Our dataset is imbalanced
- Classification accuracy is not a good performance measure
- False negatives do more harm, so maximising recall is important

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F2 Score & ROC AUC Score

- F2 Score can emphasize the recall, minimizing false negatives

$$F_2 = \frac{(1+2^2) \cdot \text{Precision} \cdot \text{Recall}}{2^2 \cdot \text{Precision} + \text{Recall}}$$

- ROC (Receiver Operating Characteristic) Curve tells us about how good the model can distinguish between two things

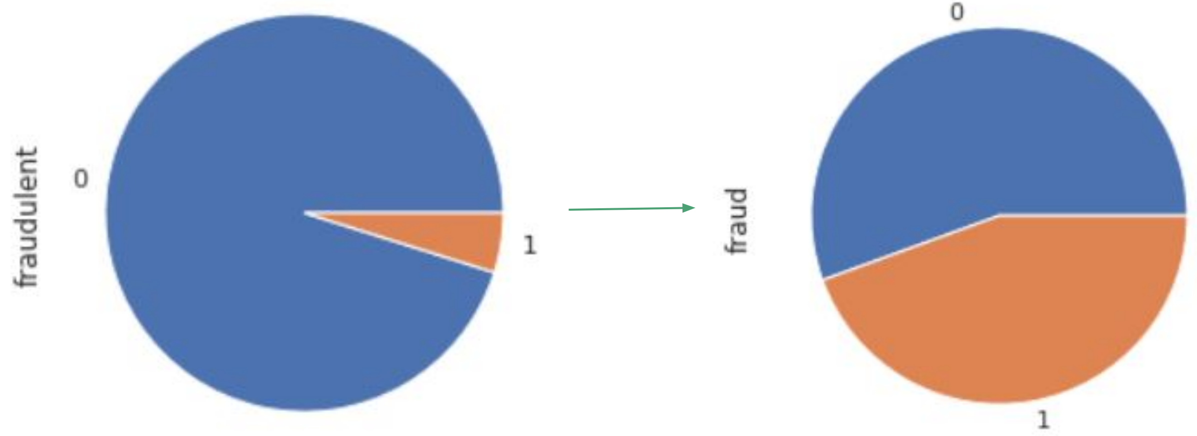
* Fake postings do more harm then missing out on the real postings!

Oversampling

```
oversample = SMOTE(sampling_strategy=0.8, random_state=0)
X_train_vectorized, y_train = oversample.fit_resample(X_train_vectorized, y_train)
```

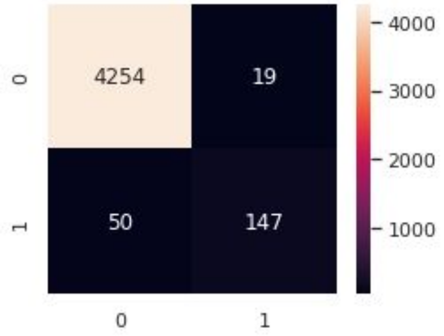
- Correct the imbalanced dataset
- Not applied on test set

```
Percentage of fraudulent job postings for train set = 44.44 %
```



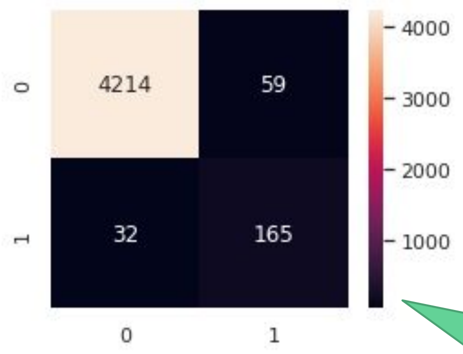
BEFORE

ROC AUC : 0.8709
F2 Score : 0.7704



	0	1
0	4254	19
1	50	147

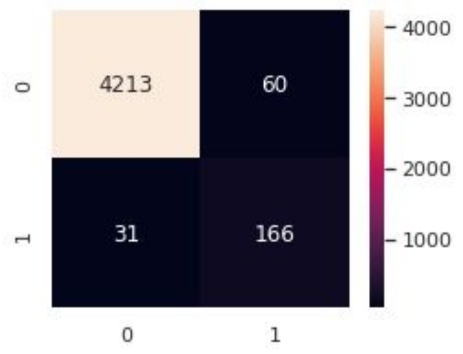
SMOTE



ROC AUC:
0.9119

F2 Score:
0.8152

SVMSMOTE

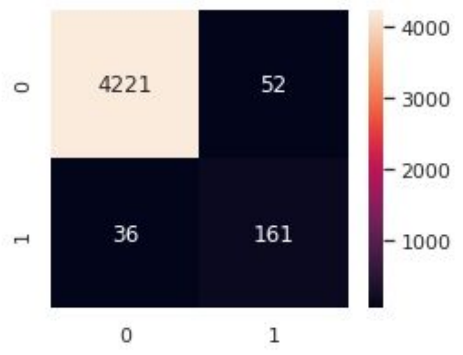


ROC AUC:
0.9143

F2 Score:
0.8185

We choose this as SVMSMOTE is very slow, and decent F2 score.

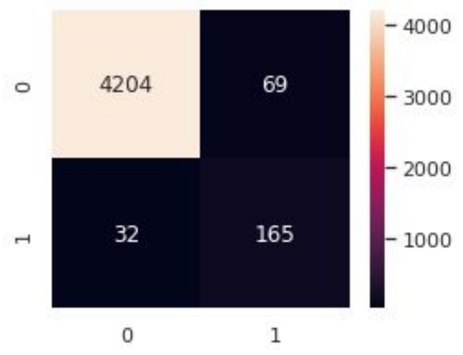
BorderlineSMOTE



ROC AUC:
0.9025

F2 Score:
0.8042

ADASYN



ROC AUC:
0.9107

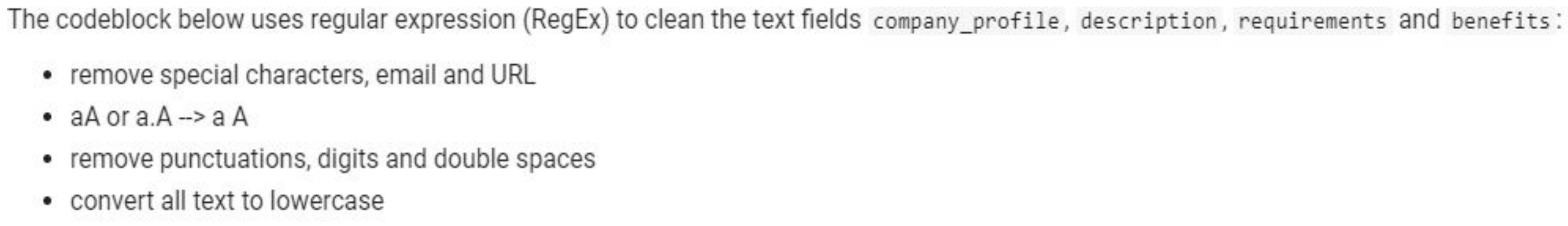
F2 Score:
0.8072

Pre-processing

Feature extraction

1. telecommuting :	0 -> telecomuting_no 1 -> telecomuting_yes
2. required experience:	Internship -> required_experience_internship
3. Fill blank with '_'	required_education_ bachelor's degree -> required_education_bachalor's_degree
4. fillna	
5. lowercase:	country_US -> country_us

Text Cleaning


The codeblock below uses regular expression (RegEx) to clean the text fields `company_profile`, `description`, `requirements` and `benefits`:

- remove special characters, email and URL
- aA or a.A --> a A
- remove punctuations, digits and double spaces
- convert all text to lowercase

Before	After
ProgramAssisting 32 in day-to-day	program assisting in day to day
WEBSITE is #URL_b0bc289	website is

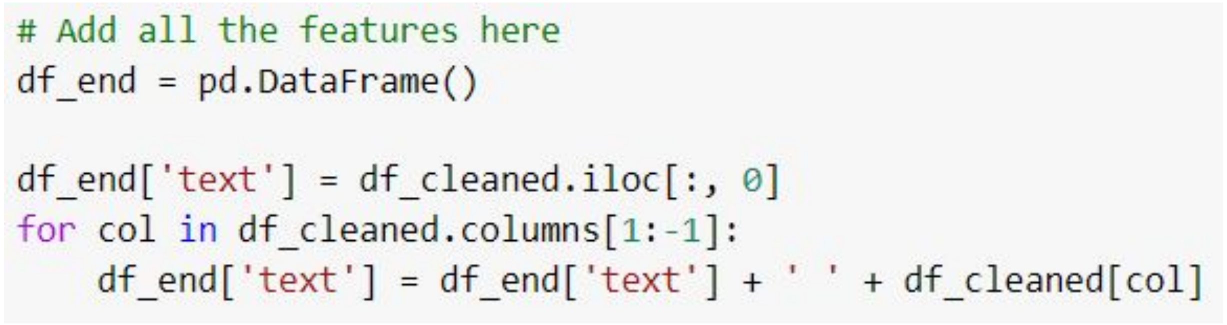
Remove Stopwords

Stopwords are common words like "I, the, an" which needs to be removed as they will be counted in during the vocabulary frequency.

Before	**After**
this position is high paying	position high paying
you will be paid weekly via direct	paid weekly direct

Combine Text Feature

Combine all columns after feature extraction



```
# Add all the features here
df_end = pd.DataFrame()

df_end['text'] = df_cleaned.iloc[:, 0]
for col in df_cleaned.columns[1:-1]:
    df_end['text'] = df_end['text'] + ' ' + df_cleaned[col]
```

Sample text:

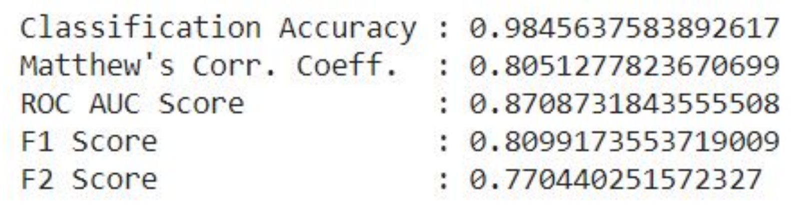
```
telecommuting_no country_us state_ny city_new york company_logo_yes
company_profile_yes has_questions_no employment_type_other
required_experience_internship function_marketing marketing intern
department_marketing food weve created groundbreaking award winning cooking site
support connect celebrate home cooks
```

Lemmatization

Change a word to its base form Eg. hearing ➔ hear ; management ➔ manage

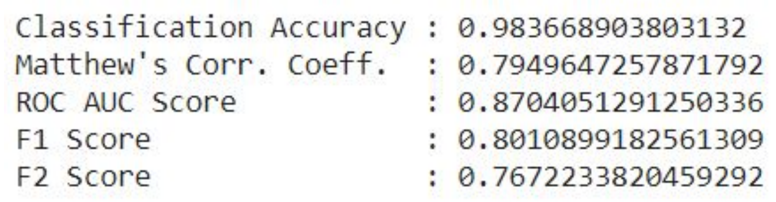
- Purpose: To reduce the vocabulary count for similar word meanings
- Downside: Takes a long time as you need to map POS tag for each word

* But this reduced our metric scores, so we won’t use it!



```
Classification Accuracy : 0.9845637583892617
Matthew's Corr. Coeff.  : 0.8051277823670699
ROC AUC Score           : 0.8708731843555508
F1 Score                : 0.8099173553719009
F2 Score                : 0.770440251572327
```

Before (All text features combined)



```
Classification Accuracy : 0.983668903803132
Matthew's Corr. Coeff.  : 0.7949647257871792
ROC AUC Score           : 0.8704051291250336
F1 Score                : 0.8010899182561309
F2 Score                : 0.7672233820459292
```

After (All text features + lemmatization)

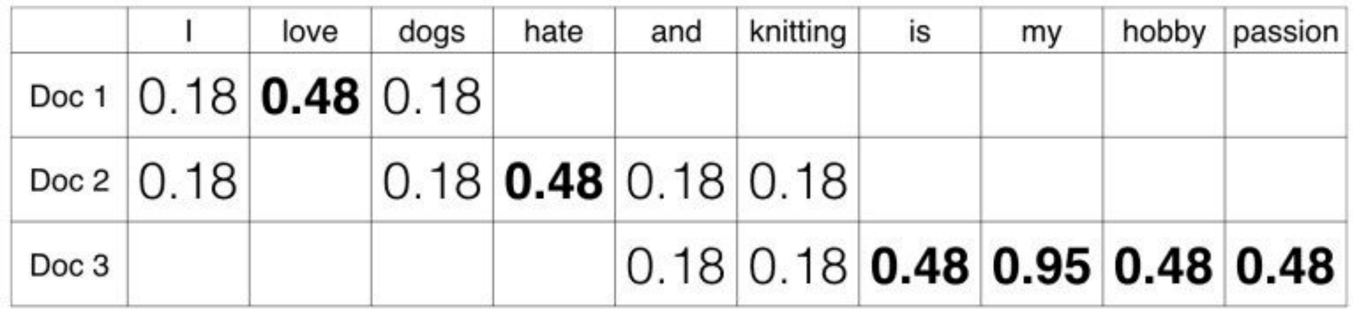
Modelling - Sentiment Analysis

Random Forest Classifier, Logistic Regression, Bernoulli Naive Bayes, Multinomial Naive Bayes, Support Vector Machine, Simple Neural Network

TF-IDF Vectorizer

TF-IDF: term frequency–inverse document frequency

- Only use a max of 20000 text features used, if all features are used:
  - Takes more time & lower metric scores



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	0.18	**0.48**	0.18							
Doc 2	0.18		0.18	**0.48**	0.18	0.18				
Doc 3					0.18	0.18	**0.48**	**0.95**	**0.48**	**0.48**

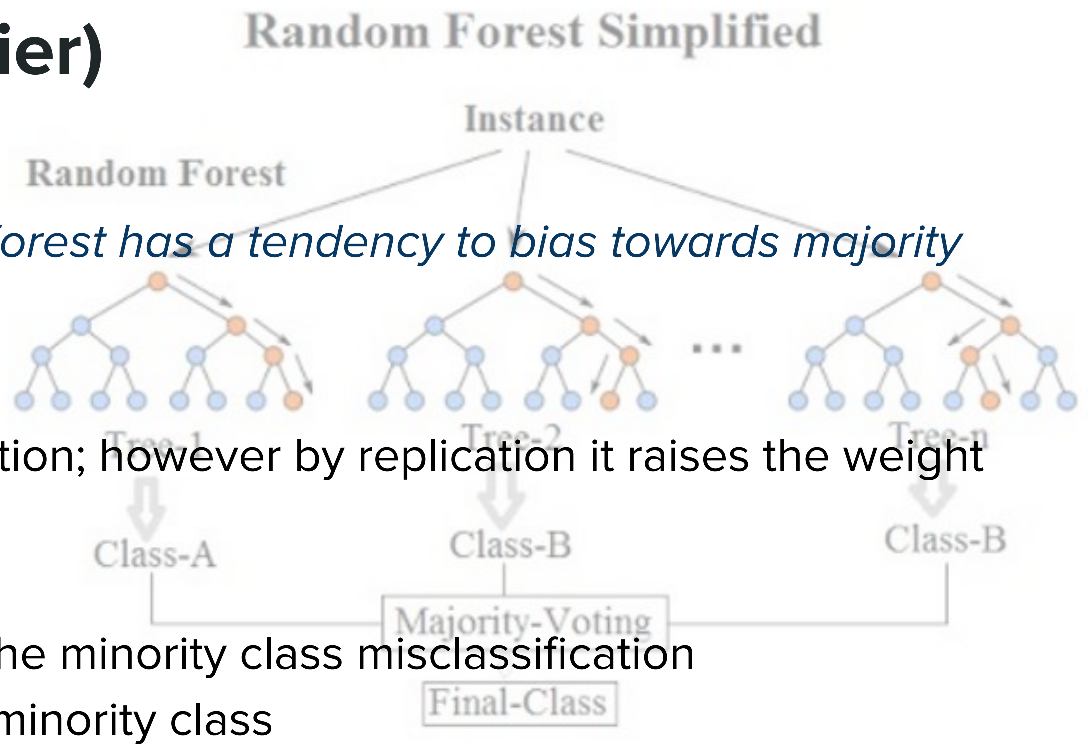
- Unigram and bigrams are both used

Modelling

1. Random Forest Classifier

- Grid Search CV

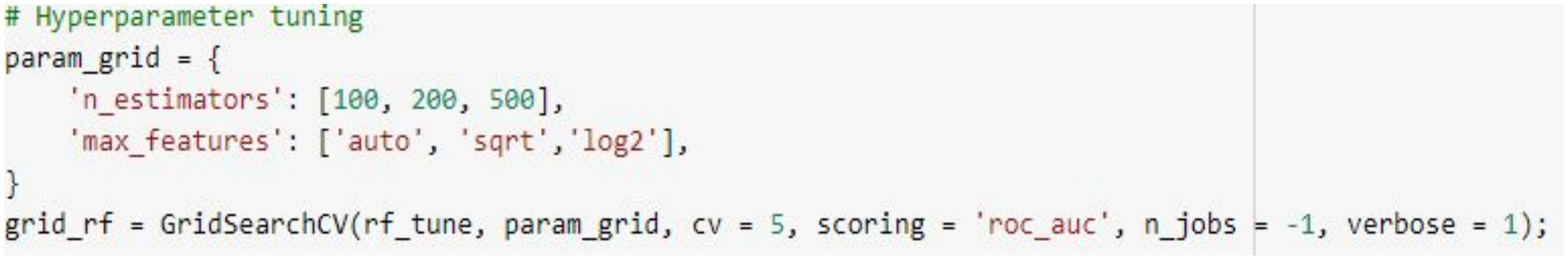
Correct Imbalance Datasets (RandomForestClassifier)



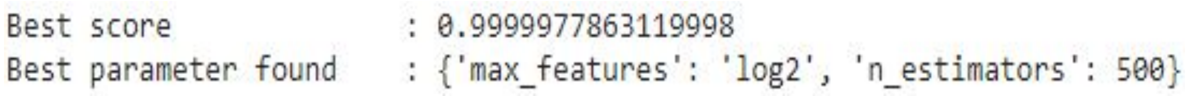
Problem:
Imbalance dataset and RandomForest has a tendency to bias towards majority class.

1. Oversampling
   - Does not increase information; however by replication it raises the weight of the minority samples.
2. Class weighting
   - Imposts a cost penalty to the minority class misclassification
   - Assign high weight to the minority class
   - set class_weight="balanced"

GridSearchCV



```
# Hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 500],
    'max_features': ['auto', 'sqrt','log2'],
}
grid_rf = GridSearchCV(rf_tune, param_grid, cv = 5, scoring = 'roc_auc', n_jobs = -1, verbose = 1);
```



```
Best score              : 0.9999977863119998
Best parameter found    : {'max_features': 'log2', 'n_estimators': 500}
```

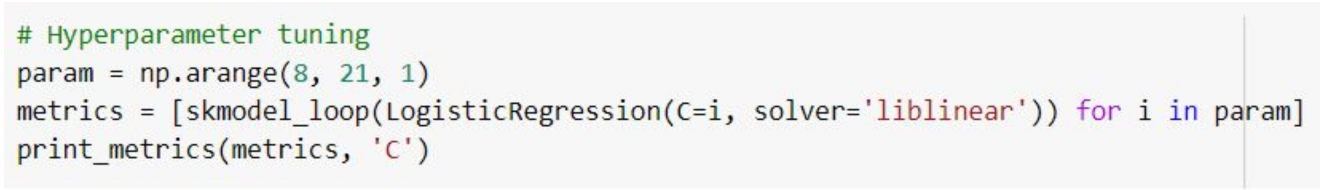
- GridSearchCV is a library function in sklearn.model_selection package to autonomate the process of performing hyperparameter tuning.
- It tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method in order to determine the hyperparameter value set which provides the best ROC AUC score (scoring = 'roc_auc')
- 5-fold cross validation is applied in this case.

Modelling

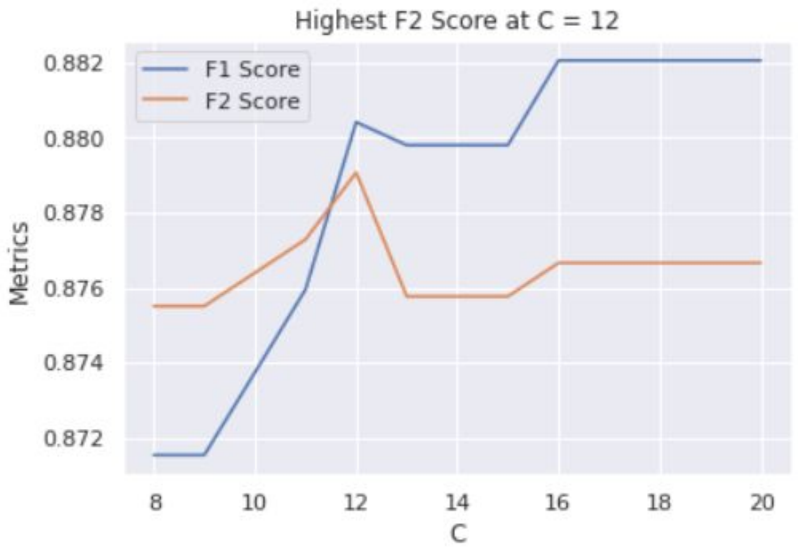
2. Logistic Regression
3. Bernoulli Naive Bayes
4. Multinomial Naive Bayes
5. Support Vector Machine

- Hyperparameter tuning

Hyperparameter Tuning



```
# Hyperparameter tuning
param = np.arange(8, 21, 1)
metrics = [skmodel_loop(LogisticRegression(C=i, solver='liblinear')) for i in param]
print_metrics(metrics, 'C')
```



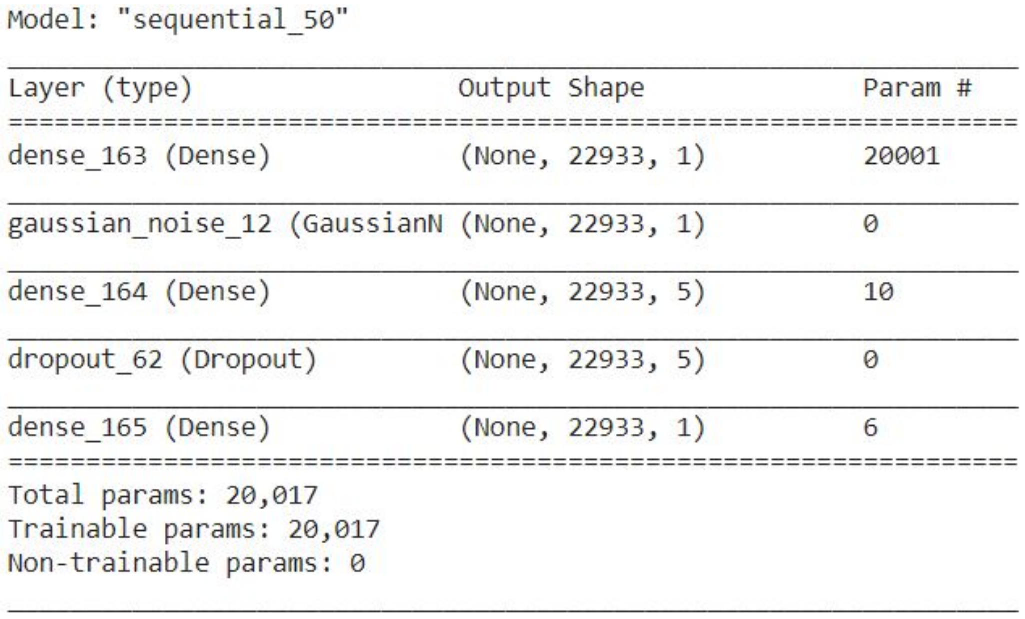
Custom functions are used to tune hyperparameters for each model used

Modelling

6. Simple Neural Network (Keras)

- Custom layers

Simple Neural Network



```
Model: "sequential_50"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_163 (Dense)            (None, 22933, 1)          20001
_________________________________________________________________
gaussian_noise_12 (GaussianN (None, 22933, 1)          0
_________________________________________________________________
dense_164 (Dense)            (None, 22933, 5)          10
_________________________________________________________________
dropout_62 (Dropout)         (None, 22933, 5)          0
_________________________________________________________________
dense_165 (Dense)            (None, 22933, 1)          6
=================================================================
Total params: 20,017
Trainable params: 20,017
Non-trainable params: 0
_________________________________________________________________
```

Gaussian Noise and Dropout layers are added to prevent overfitting!

Custom callback code made for this section.

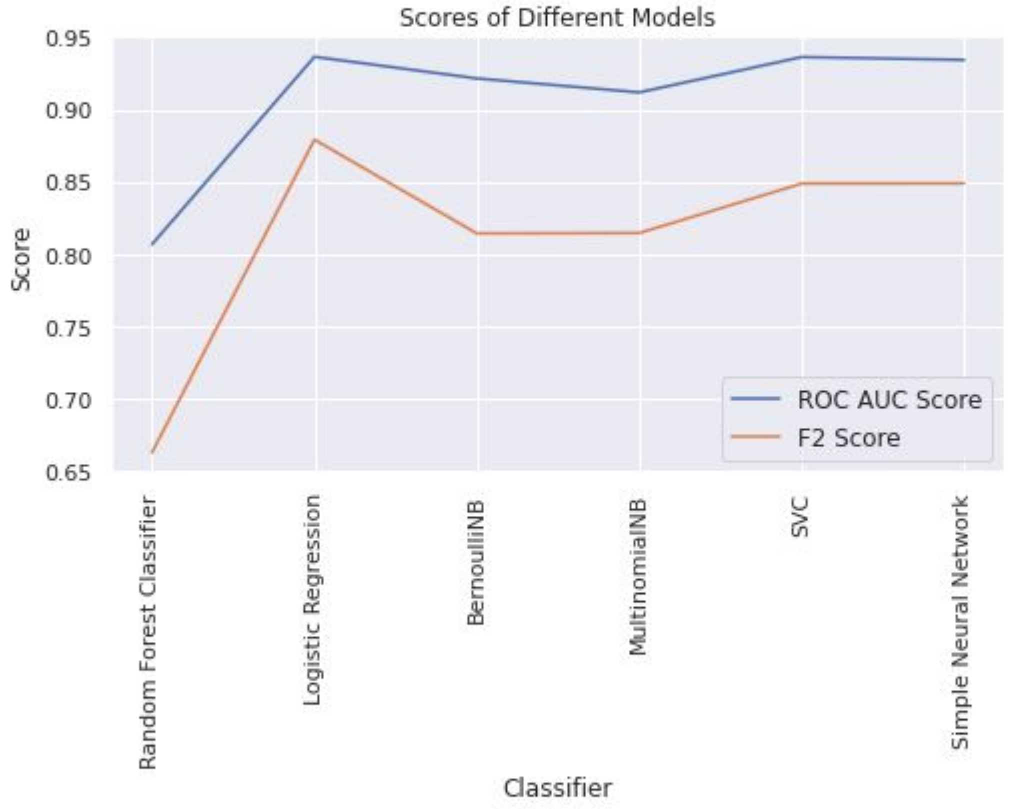
- Monitors F2 Score

Comparing all models!

Logistic Regression is the best model!

✓ *Which model is the best?*

!! No experience in deep learning
--> Overfitting

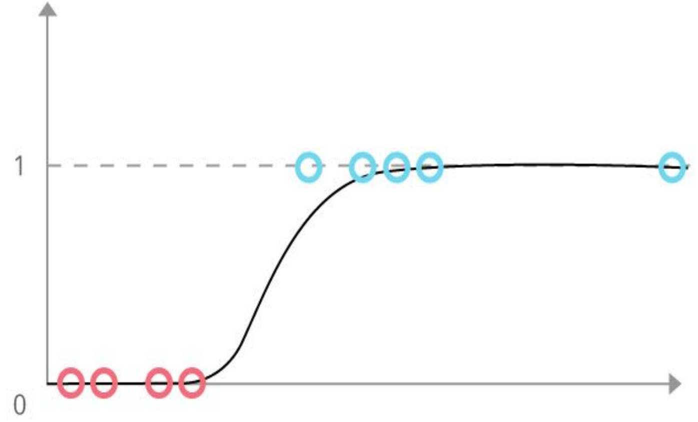


Logistic Regression

Logistic Regression can be used as a classification model

- Performs linear regression first, output is ***z***
- Then passes through a sigmoid function, ***y*** = σ(***z***)
- Distinct boundary at ***y*** = 0.5
  (above is 1, below is 0)

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



- Is a form of neural network, but scikit-learn takes care of the technical implementation

Logistic Regression (pt. 2)

Default L2 regularization is used to prevent overfitting

- An additional term is added to the cost function of the model
- Model will try to minimise this cost function

L2 Regularization

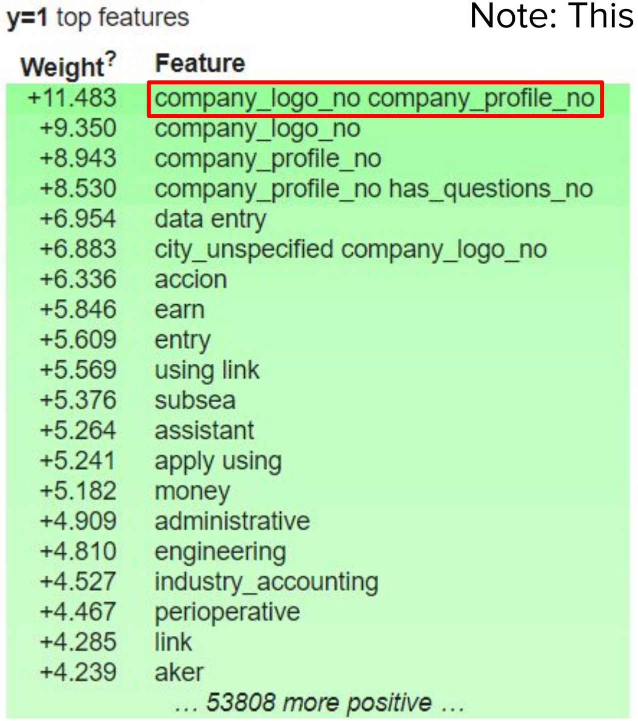
$$\text{Cost} = \underbrace{\sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Parameter ‘C’ - is the inverse regularization strength

Top Features

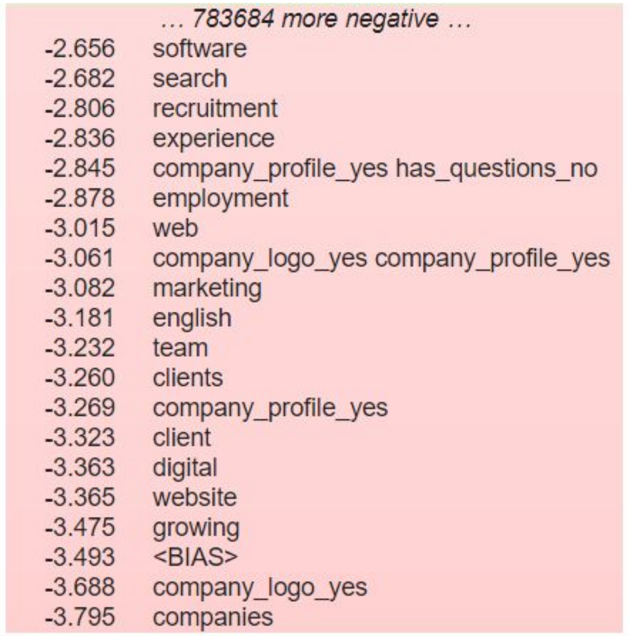
```
eli5.explain_weights(trained_model5, feature_names=vect.get_feature_names(), top=(20, 20))
```

Note: This is for all 837531 features! (not just 20000)



y=1 top features

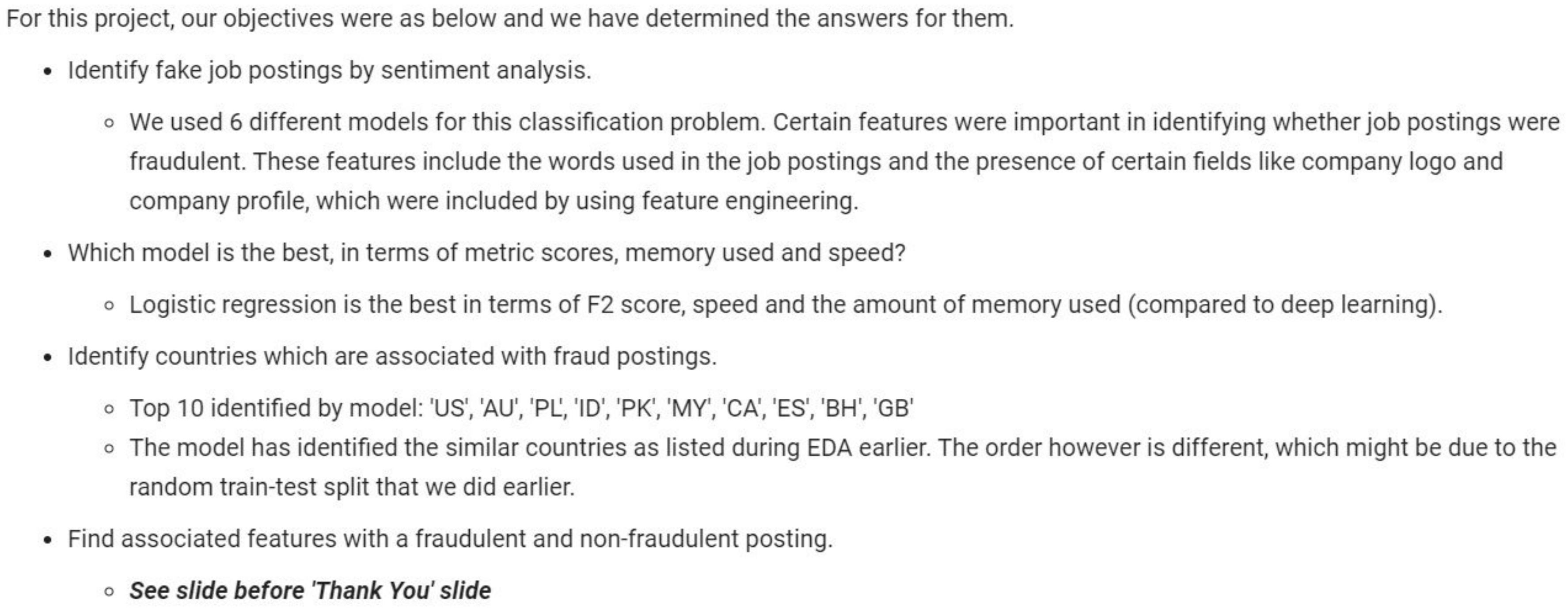
Weight?	Feature
+11.483	company_logo_no company_profile_no
+9.350	company_logo_no
+8.943	company_profile_no
+8.530	company_profile_no has_questions_no
+6.954	data entry
+6.883	city_unspecified company_logo_no
+6.336	accion
+5.846	earn
+5.609	entry
+5.569	using link
+5.376	subsea
+5.264	assistant
+5.241	apply using
+5.182	money
+4.909	administrative
+4.810	engineering
+4.527	industry_accounting
+4.467	perioperative
+4.285	link
+4.239	aker
	… 53808 more positive …



	… 783684 more negative …
-2.656	software
-2.682	search
-2.806	recruitment
-2.836	experience
-2.845	company_profile_yes has_questions_no
-2.878	employment
-3.015	web
-3.061	company_logo_yes company_profile_yes
-3.082	marketing
-3.181	english
-3.232	team
-3.260	clients
-3.269	company_profile_yes
-3.323	client
-3.363	digital
-3.365	website
-3.475	growing
-3.493	<BIAS>
-3.688	company_logo_yes
-3.795	companies

Thank you

A summary of our takeaways



For this project, our objectives were as below and we have determined the answers for them.

- Identify fake job postings by sentiment analysis.
    - We used 6 different models for this classification problem. Certain features were important in identifying whether job postings were fraudulent. These features include the words used in the job postings and the presence of certain fields like company logo and company profile, which were included by using feature engineering.
- Which model is the best, in terms of metric scores, memory used and speed?
    - Logistic regression is the best in terms of F2 score, speed and the amount of memory used (compared to deep learning).
- Identify countries which are associated with fraud postings.
    - Top 10 identified by model: 'US', 'AU', 'PL', 'ID', 'PK', 'MY', 'CA', 'ES', 'BH', 'GB'
    - The model has identified the similar countries as listed during EDA earlier. The order however is different, which might be due to the random train-test split that we did earlier.
- Find associated features with a fraudulent and non-fraudulent posting.
    - ***See slide before 'Thank You' slide***

Contributions (in Alphabetical Order)

Goh Lee Hua	Text pre-processing, Random forest classifier and GridSearchCV, Markdown comments
Hansel Tay	Lemmatization, Metrics, Oversampling and undersampling techniques
Philip Lee Hann Yung	Feature extraction, TF-IDF vectorization, Modelling and Hyperparameter tuning, Organization of project pipeline, Markdown comments
Tan Keng Soon	Visualisation, Exploratory data analysis (EDA)

Q & A