



IDENTIFYING FAKE JOB POSTINGS

By:
Charles Albert
Hilda Flores
Caroline Kurzweg

TABLE OF CONTENTS

01

PROJECT OVERVIEW

Brief, success of the project and assumptions.

02

DATASET

Obtaining, wrangling and creation of training and test dataset.

03

MODELLING

Selection of best model.
Prediction score and confusion matrix.

04

INSIGHTS

Interesting findings and lessons learned.

05

QUESTIONS

Section of questions and answers.



01

PROJECT OVERVIEW

FAKE JOB POSTINGS

When people are looking through different sites to find the most interesting job postings according to their professional objectives, they may encounter various job postings and send personal information such as resumé, contact information and even social security number to further proceedings regarding job interviews.

Nonetheless, though rare, there is a risk that the job posting might be fake. Consequently, personal information is at stake.

Therefore, this project aims to identify fake job postings in order to reduce the risk of being the victim of fraud and potentially identity theft .



How Will We Determine Success?

- 95% of data is as not fraudulent.
- Accuracy > 95% will be considered a success.
- Small amount of success is still important because personal information is at stake.
- We would rather false positives due to the serious nature of the problem.

ASSUMPTIONS

Variables
measured at
a nominal
level

Data set is
representative
of the whole
population

Variables
consist of two
or more
categorical
independent
groups

Missing values
indicate that they
were not included
in the job listing


```
def __init__(self, path):
    self.file = None
    self.fingerprints = set()
    self.logdups = True
    self.debug = debug
    self.logger = logging.getLogger(__name__)
    if path:
        self.file = open(os.path.join(path, 'fingerprints.log'), 'a')
        self.file.seek(0)
        self.fingerprints.update(set(self.file.read().splitlines()))

    @classmethod
    def from_settings(cls, settings):
        debug = settings.DEBUG
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + '\n')

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```

02

DATASET

FAKE JOB DATA

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17880 entries, 0 to 17879
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   job_id                17880 non-null  int64
 1   title                 17880 non-null  object
 2   location              17534 non-null  object
 3   department            6333 non-null   object
 4   salary_range          2868 non-null   object
 5   company_profile       14572 non-null  object
 6   description           17879 non-null  object
 7   requirements          15185 non-null  object
 8   benefits              10670 non-null  object
 9   telecommuting         17880 non-null  int64
10   has_company_logo      17880 non-null  int64
11   has_questions         17880 non-null  int64
12   employment_type       14409 non-null  object
13   required_experience    10830 non-null  object
14   required_education    9775 non-null   object
15   industry              12977 non-null  object
16   function              11425 non-null  object
17   fraudulent            17880 non-null  int64
dtypes: int64(5), object(13)
memory usage: 2.5+ MB
```

Overview

- From The University of the Aegean
- 17880 Observations
- 18 Variables
- Dependent Variable: 'fraudulent'
- Only 1 Missing Value for Description(IV)
- Other Variables Missing Lots of Data

GOOD DATASET?

For our purposes, this data set is good. Since our Independent Variable, 'Description' , is only missing 1 value and our dependent variable, 'Fraudulent', is missing no values, we are considering this a good, clean data set.

DATA WRANGLING

```
6    description    17879 non-null object
```

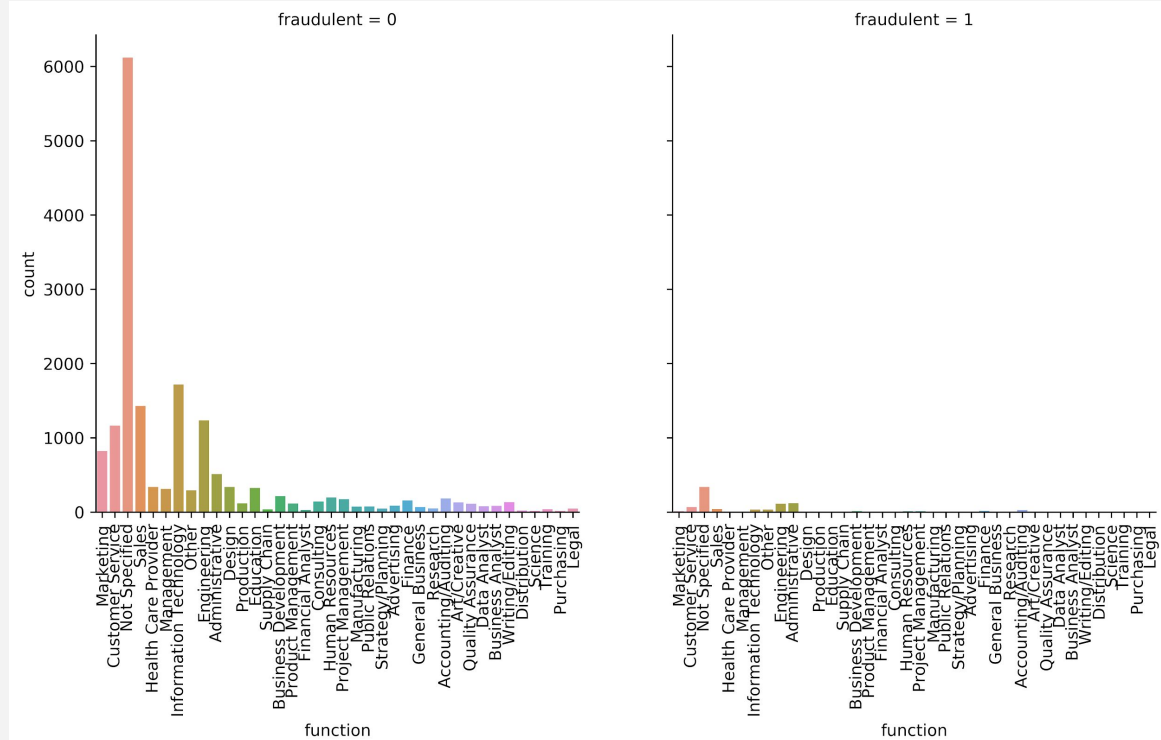
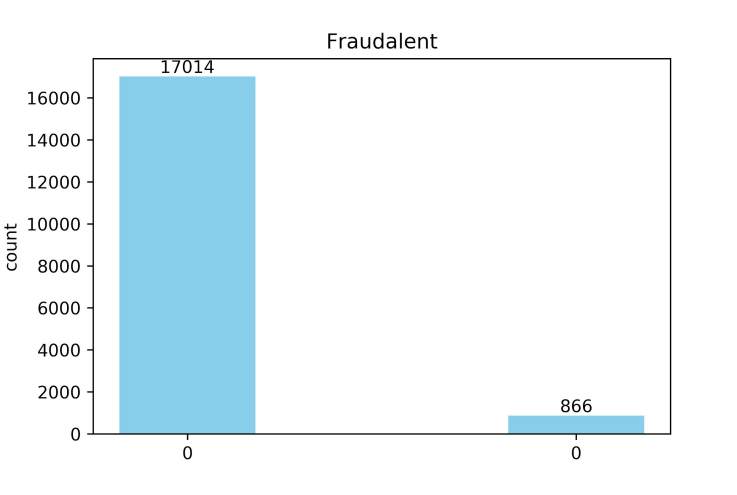
```
#replacing the NaN with blank string for NLP  
df_job.replace(np.NaN, '', inplace = True)
```

```
6    description    17880 non-null object
```

```
#dropping the columns that we are not going to run through the NLP  
df_job = df_job.drop(['location', 'department', 'salary_range', 'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',  
                     'required_experience', 'required_education', 'industry', 'function'], axis = 1)
```

- Not much Data Wrangling needed
- Replaced NaN with ''
- Only 1 missing Description
- Drop excess columns

VISUALS





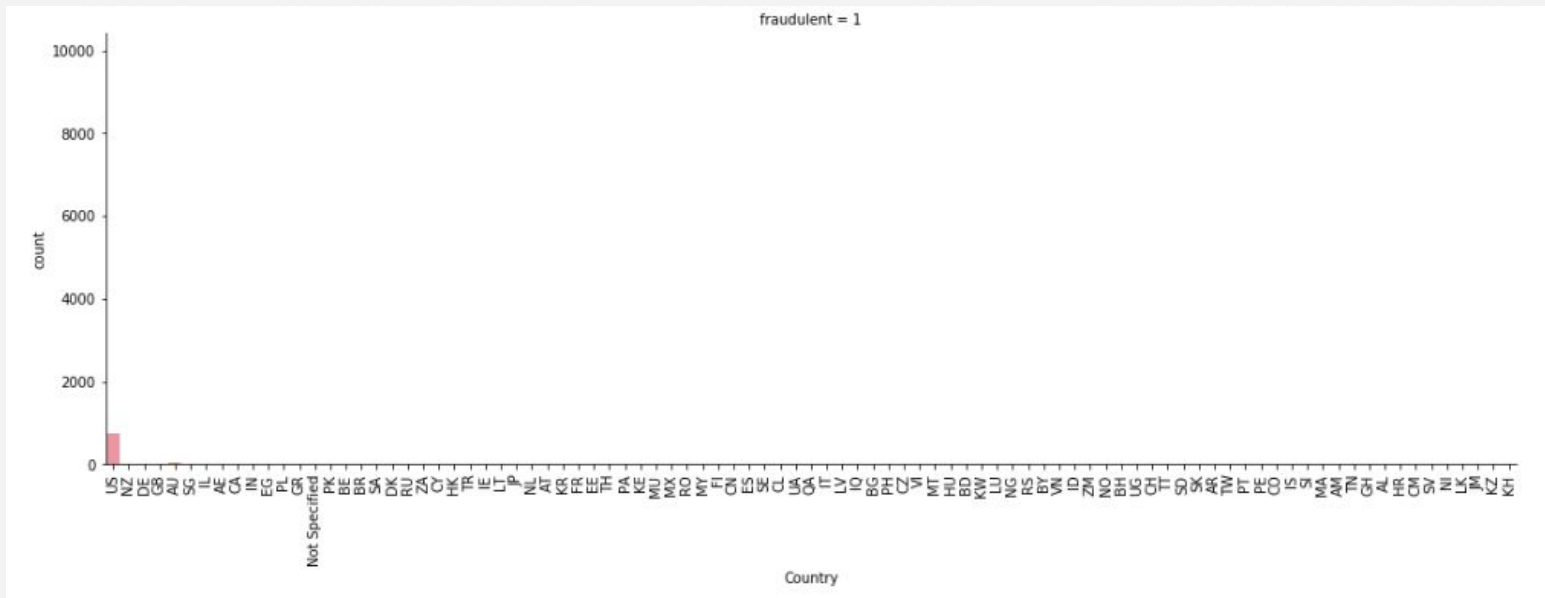
USA 1° in fraudulent job postings
84.3%



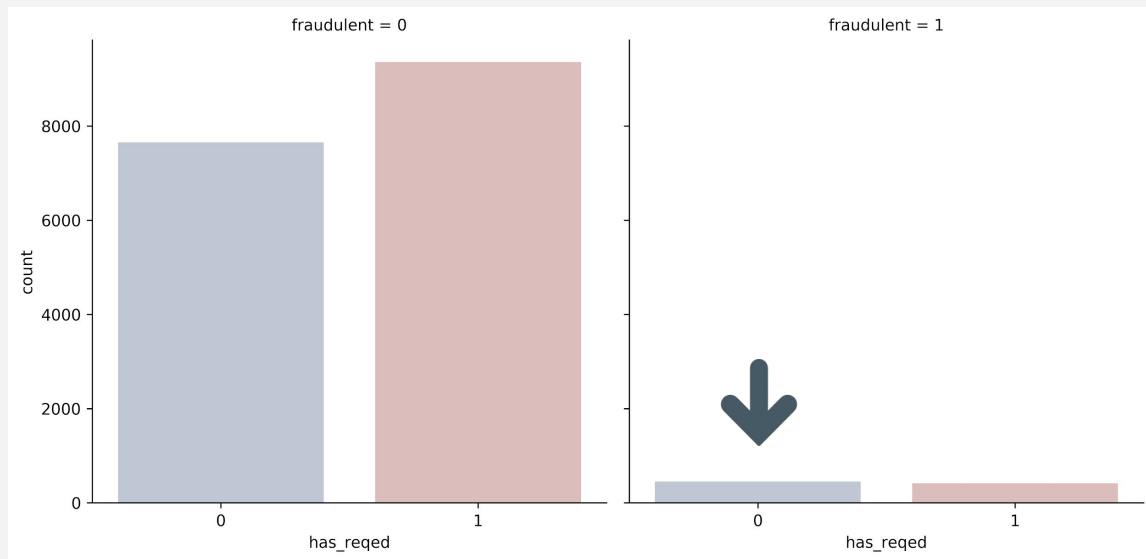
4.6%



2.7%



VISUALS



For future investigations...

More sampling of fake job postings could determine if not posting required education is key to classify.

has_reqed	0	1
fraudulent		
0	7654	9360
1	451	415

03

MODELING



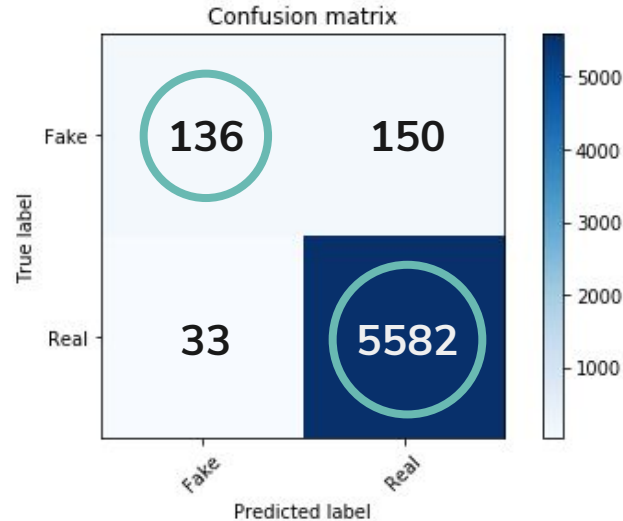
NATURAL LEARNING PROCESS

PREDICTION SCORE

.9690



Confusion matrix, without normalization



***Random state = 25

Example Code

```
X_train, X_test, y_train, y_test = train_test_split(df_job['description'], y, test_size=0.33, random_state=53, stratify = y)
```

```
# Create bag-of-word vectors for the news articles  
# https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction  
count_vectorizer = CountVectorizer(stop_words='english')  
count_train = count_vectorizer.fit_transform(X_train)  
count_test = count_vectorizer.transform(X_test)
```

```
# Instantiate a Multinomial Naive Bayes classifier: nb_classifier  
nb_classifier = MultinomialNB()
```

```
# Fit the classifier to the training data  
nb_classifier.fit(count_train, y_train)
```

```
# Create the predicted tags: pred  
pred = nb_classifier.predict(count_test)
```

```
# Calculate the accuracy score: score  
score = metrics.accuracy_score(y_test, pred)  
print(score)
```

```
# Calculate the confusion matrix: cm  
cm = metrics.confusion_matrix(y_test, pred, labels=[1,0])  
print(cm)
```

```
0.963226571767497
```

Example Code

```
x_train, x_test, y_train, y_test = train_test_split(df_job['description'], y, test_size=0.33, random_state=25, stratify = y)
```

```
# Create bag-of-word vectors for the news articles  
# https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction  
count_vectorizer = CountVectorizer(stop_words='english')  
count_train = count_vectorizer.fit_transform(X_train)  
count_test = count_vectorizer.transform(X_test)
```

```
# Instantiate a Multinomial Naive Bayes classifier: nb_classifier  
nb_classifier = MultinomialNB()
```

```
# Fit the classifier to the training data  
nb_classifier.fit(count_train, y_train)
```

```
# Create the predicted tags: pred  
pred = nb_classifier.predict(count_test)
```

```
# Calculate the accuracy score: score  
score = metrics.accuracy_score(y_test, pred)  
print(score)
```

```
# Calculate the confusion matrix: cm  
cm = metrics.confusion_matrix(y_test, pred, labels=[1,0])  
print(cm)
```

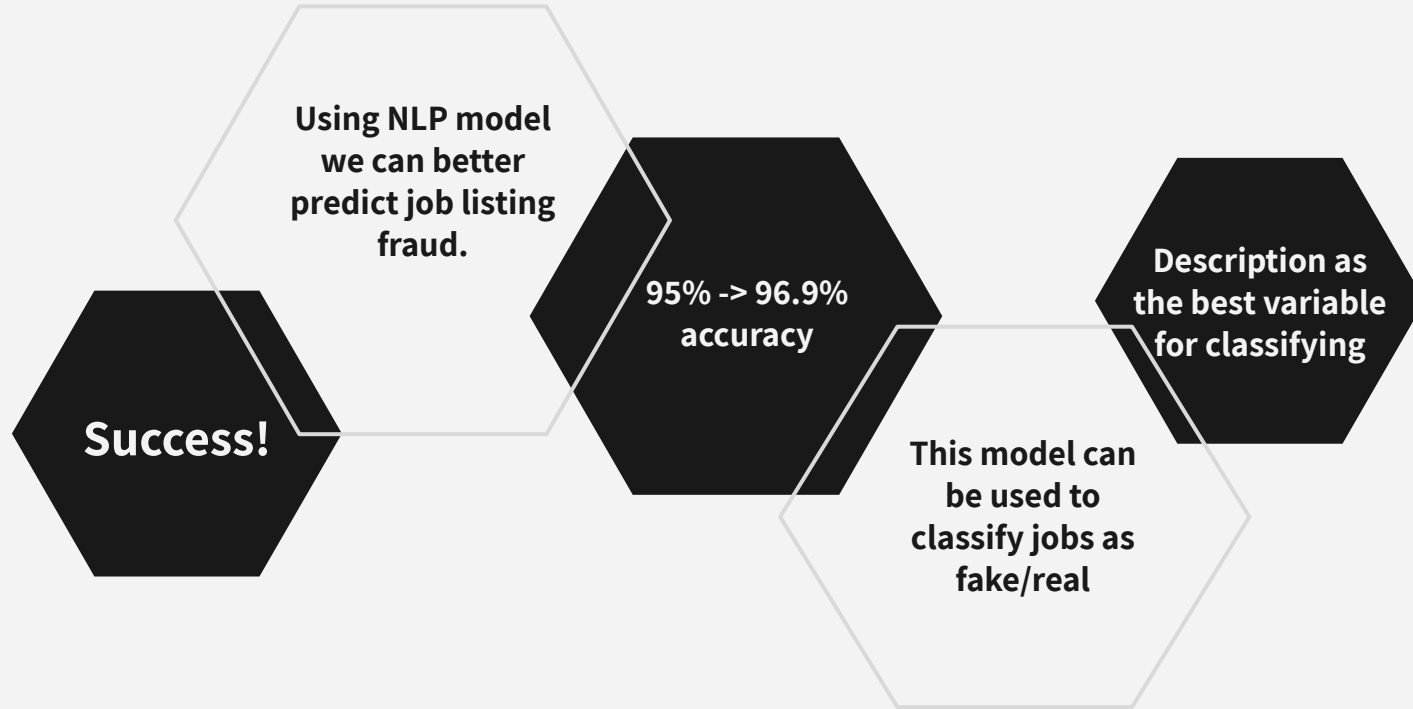
```
0.9689883070665989
```



04

INSIGHTS

Insights From Model



Other Models Tried

- NLP using Company Profile
- NLP using Requirements
- NLP using benefits
- Random Forest using dummy coded variables
- Decision Tree using dummy coded variables
- Logistic Regression



Personal Insights

- **Caroline**
 - **Get creative! There are so many ways to get interesting insights. Do the research and learn new ways to create models. Even if the models are not successful, thinking outside the box will help me learn more.**
- **Carter**
 - **I thought that it was very interesting that the benefits column was such a high performer. If done again, I would like to try to incorporate salary range to see if fraudulent job postings are advertising a certain structure to their salary and benefits package as an incentive to get people to apply.**
- **Hilda**
 - **I was very surprised that industrialized countries had the more fake job postings than the most corrupt countries in the world! Personally, I think it is due to lack of sampling on those countries. So, I would get more sampling on fake job postings to corroborate this insight. Also, I would include in which sites are the fake job postings are advertised and accounts of enterprises that contact you through different platforms. This would help visualize which platforms need more security and help identify which sites are preferred to advertise fake job postings.**



THANKS

Does anyone have any questions?