fake_job_post_1

June 9, 2020

Data cleaning and exploratory data analysis.

[2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")
%matplotlib inline
from collections import Counter
import numpy as np
pd.set_option('display.max_columns', None)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

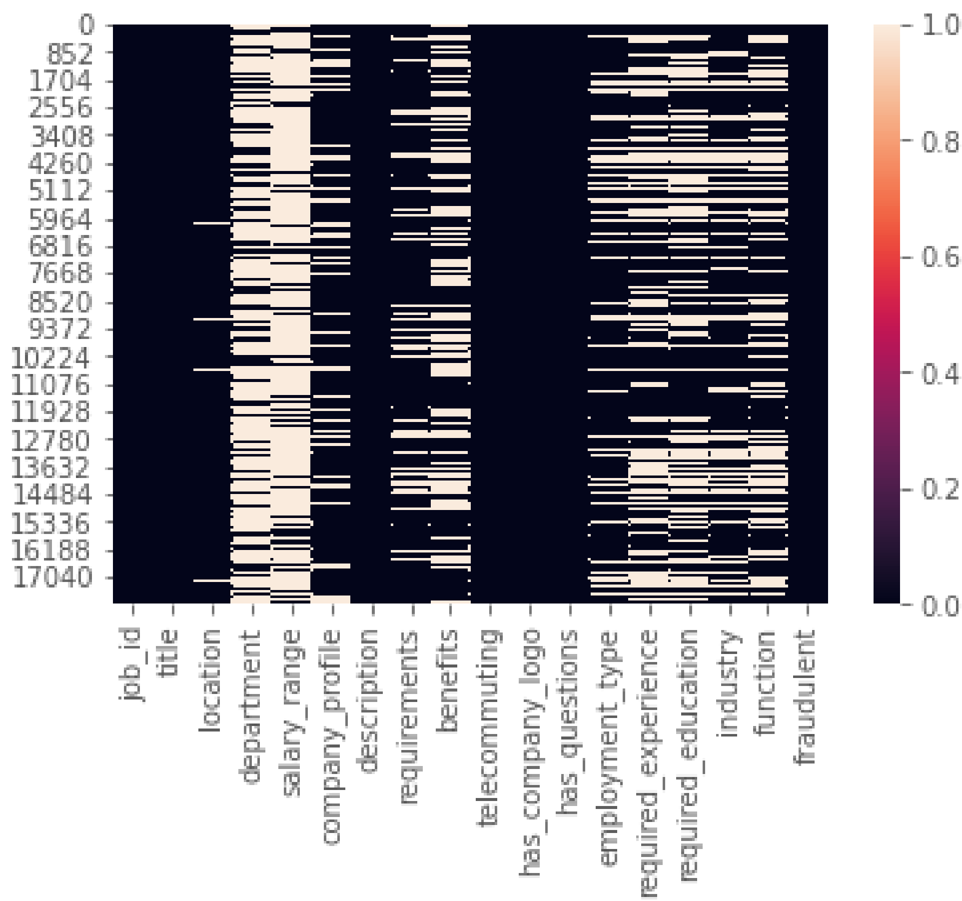
```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/drive
```

[0]:
```
df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/fake_job_postings.
↪csv')
```

[5]:
```
print(df.isna().sum())
sns.heatmap(df.isnull())
```

```
job_id                    0
title                     0
location                346
department            11547
salary_range          15012
company_profile        3308
description               1
requirements           2695
benefits               7210
telecommuting             0
has_company_logo          0
has_questions             0
employment_type        3471
required_experience    7050
required_education     8105
industry               4903
function               6455
fraudulent                0
dtype: int64
```

[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f993d807470>



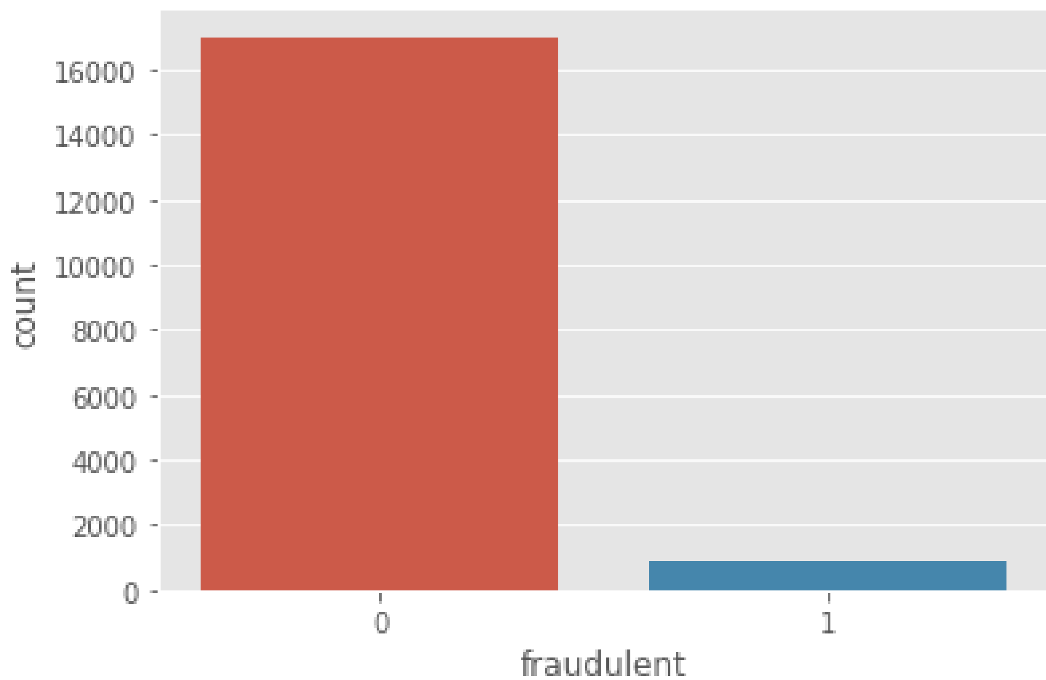
Heatmap shows data has many null values. So it needs tobe cleaned first.

[7]:
```
print(df.shape)
```

```
(17880, 18)
```

[8]:
```
plt.figure(1)
sns.countplot( x= df['fraudulent'], data= df)
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe95bf08da0>
```



Also in the main data only 6 percent values are fraudelent.

[6]:
```
df2 = df.copy()
df2.drop(['salary_range', 'job_id', 'department', 'benefits'], axis = 1,␣
 ↪inplace = True)
df2 = df2.sort_values('title').reset_index(drop = True)
df2.isna().sum()
```

[6]:
```
title                  0
location             346
company_profile     3308
description            1
requirements        2695
telecommuting          0
```

```
has_company_logo          0
has_questions             0
employment_type        3471
required_experience    7050
required_education     8105
industry               4903
function               6455
fraudulent                0
dtype: int64
```

Some features are dropped
Job id = This is not useful in prediction.
Benefits = It can be covered in other features.

---

[0]:
```python
df2['employment_type'] = df2['employment_type'].bfill(axis=0)
df2['required_experience'] = df2['required_experience'].bfill(axis=0)
df2['required_education'] = df2['required_education'].bfill(axis = 0)
df2['industry'] = df2['industry'].bfill(axis=0)
df2['function'] = df2['function'].bfill(axis=0)
```

Features like these can be back filled because many of them have less no of categories. And even if there occurs a mismatch due to bfill it will be in less proportion and will not affect the prediction.

[0]:
```python
df3 = df2.copy()
```

[0]:
```python
df3 = df3[df3['description'].notna()]
```

[10]:
```python
print(df3.isna().sum())
print(df3.shape)
```

```
title                     0
location                346
company_profile        3307
description               0
requirements           2694
telecommuting             0
has_company_logo          0
has_questions             0
employment_type           2
required_experience       2
required_education        2
industry                  2
function                  2
fraudulent                0
dtype: int64
(17879, 14)
```

[11]:
```
df3 = df3.dropna(axis = 0, how = 'any')
df3.isna().sum()
```

[11]:
```
title                  0
location               0
company_profile        0
description            0
requirements           0
telecommuting          0
has_company_logo       0
has_questions          0
employment_type        0
required_experience    0
required_education     0
industry               0
function               0
fraudulent             0
dtype: int64
```

And with remaining features those we couldnt fill, their respective rows are simply dropped from data.

[12]:
```
df3.shape
```

[12]:
```
(12501, 14)
```

[0]:
```
df3 = df3.drop_duplicates(keep = 'first')
```

[0]:
```
df4 = df3.copy()
```

[15]:
```
df4.shape
```

[15]:
```
(12264, 14)
```

[0]:
```
df4['description'] = df4['description'] + ' ' + df4['requirements'] + ' ' +␣
 ↪df4['company_profile']
df4.drop(['company_profile', 'requirements'], axis = 1, inplace = True)
```

After this, for the ease of NLP, features which have sentences and paragraphs are concatenated to a one single feature.

[0]:
```
df4['country_code'] = df4['location'].str.split(',', expand=True)[0]
df4['city'] = df4['location'].str.split(',', expand = True)[2]
```

Country and city are separated from location.

[0]:
```
df4.loc[df4['city'] == ' ', 'city'] = np.nan
```

[19]:
```
df4.isnull().sum()
```

[19]:
```
title                  0
location               0
description            0
telecommuting          0
has_company_logo       0
```

```
has_questions            0
employment_type          0
required_experience      0
required_education       0
industry                 0
function                 0
fraudulent               0
country_code             0
city                   992
dtype: int64
```

[0]:
```
df4.dropna(inplace = True)
```

[21]:
```
pip install pycountry
```

```
Collecting pycountry
  Downloading https://files.pythonhosted.org/packages/16/b6/154fe93072051d
8ce7bf197690957b6d0ac9a21d51c9a1d05bd7c6fdb16f/pycountry-19.8.18.tar.gz (10.0MB)
     || 10.0MB 2.6MB/s
Building wheels for collected packages: pycountry
  Building wheel for pycountry (setup.py) ... done
  Created wheel for pycountry: filename=pycountry-19.8.18-py2.py3-none-any.whl
size=10627361
sha256=7acbb87b7cc0283f1afcbff9334ed960b67f4206ea9452ee73896b64901ed412
  Stored in directory: /root/.cache/pip/wheels/a2/98/bf/f0fa1c6bf8cf2cbdb750d583
f84be51c2cd8272460b8b36bd3
Successfully built pycountry
Installing collected packages: pycountry
Successfully installed pycountry-19.8.18
```

[0]:
```
import pycountry
list_alpha_2 = [i.alpha_2 for i in list(pycountry.countries)]
def country(df):
    if df['country_code'] in list_alpha_2:
        return pycountry.countries.get(alpha_2 = df['country_code']).name
df4['country_name'] = df4.apply(country, axis = 1)
```

[0]:
```
df4.drop(['location', 'country_code'], axis = 1, inplace = True)
```

[27]:
```
df4.head()
```

[27]:
```
                                               title  \
2                           Piping Material Engineer
3     Discipline Manager Civil, Structural, Marine...
4                                FEA Senior engineer
9                                    AUTOCAD OPERATOR
13                                  Accounting Clerk

                                         description  telecommuting  \
2   Corporate overviewAker Solutions is a global p...              0
```

```
3   Corporate overviewAker Solutions is a global p...              0
4   Corporate overviewAker Solutions is a global p...              0
9   Responsibilities:Using a project database syst...              0
13  Job DescriptionVerify, obtain approvals and pa...              0

    has_company_logo  has_questions employment_type required_experience  \
2                  1              0       Full-time    Mid-Senior level
3                  1              0       Full-time         Entry level
4                  1              0       Full-time         Entry level
9                  1              0       Full-time    Mid-Senior level
13                 1              1       Full-time           Associate

           required_education                 industry          function  \
2             Master's Degree             Oil & Energy       Engineering
3                Professional             Oil & Energy       Engineering
4             Master's Degree             Oil & Energy       Engineering
9           Bachelor's Degree  Staffing and Recruiting       Engineering
13  High School or equivalent               Accounting  Customer Service

    fraudulent      city   country_name
2            1   Houston  United States
3            1   Houston  United States
4            1   Houston  United States
9            0      Cebu    Philippines
13           1    AUSTIN  United States
```

[24]:
```
df4.shape
```

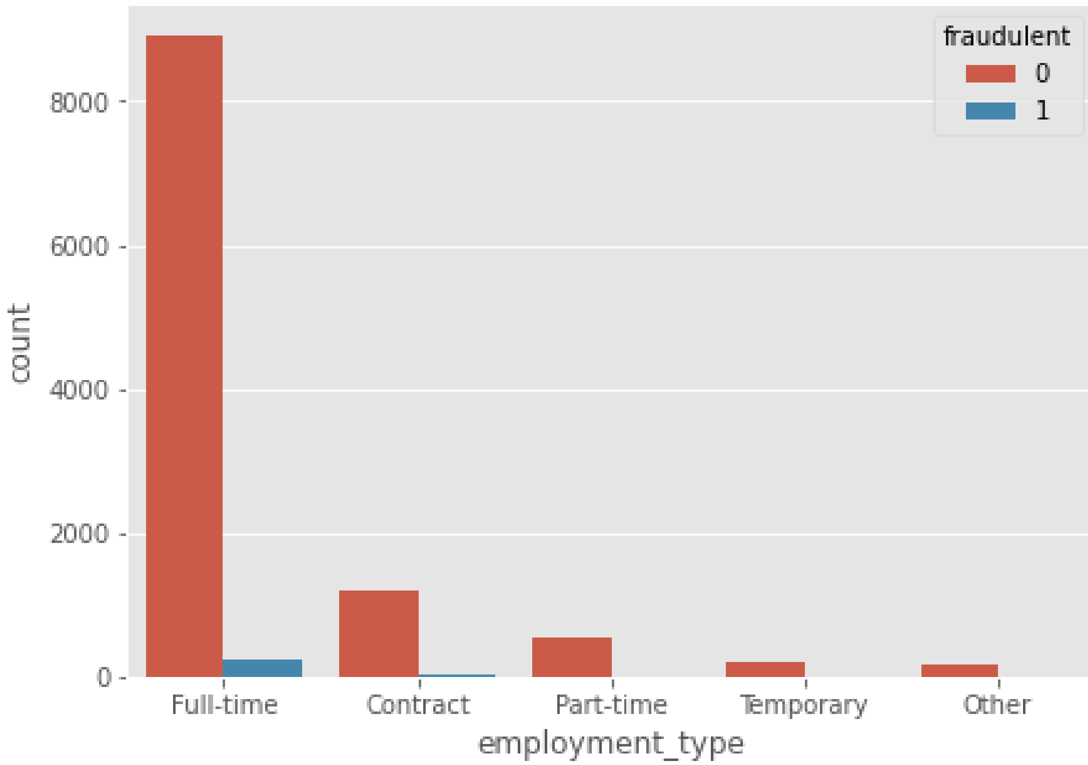
[24]: (11272, 13)

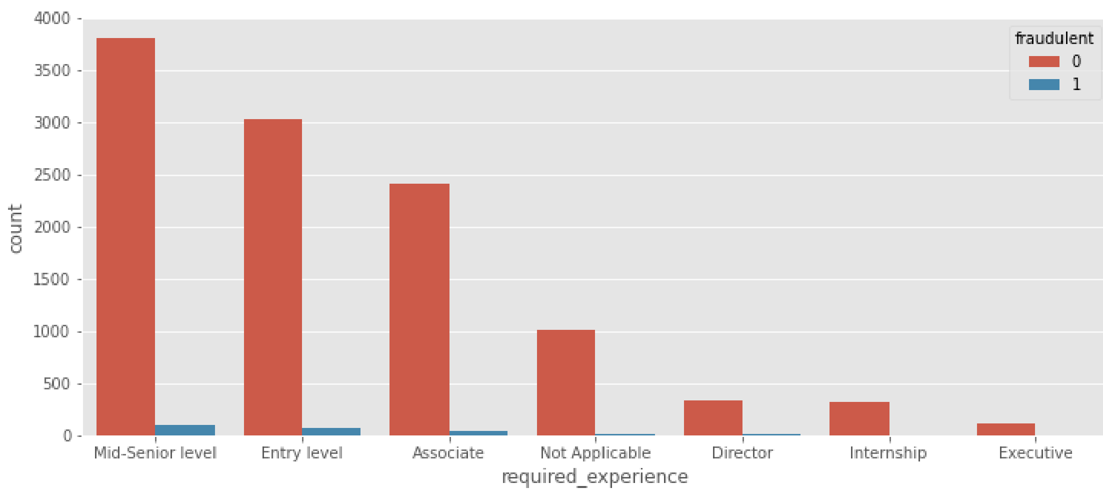
[0]:
```
df_clean = df4.copy()
```

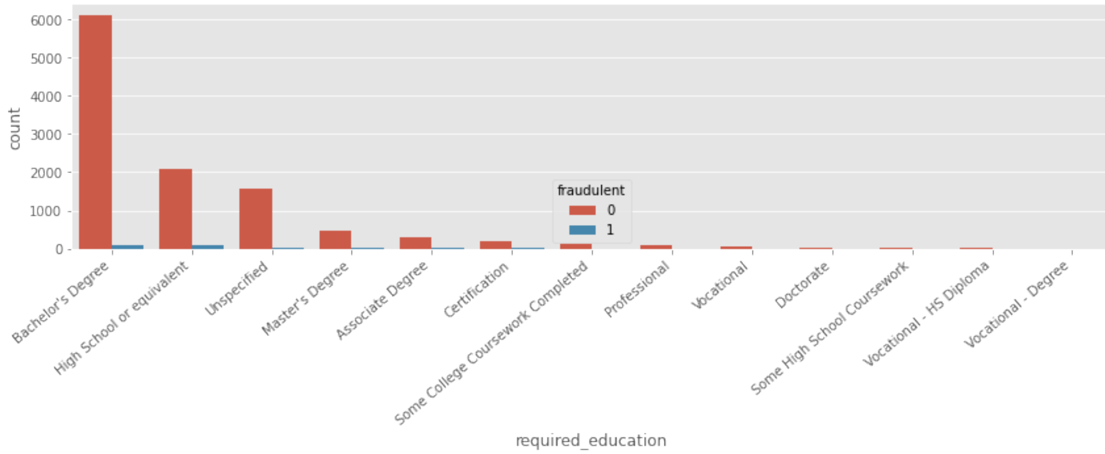
[0]:
```
df_clean.head()
df_clean.to_csv('Clean_data1.csv')
```

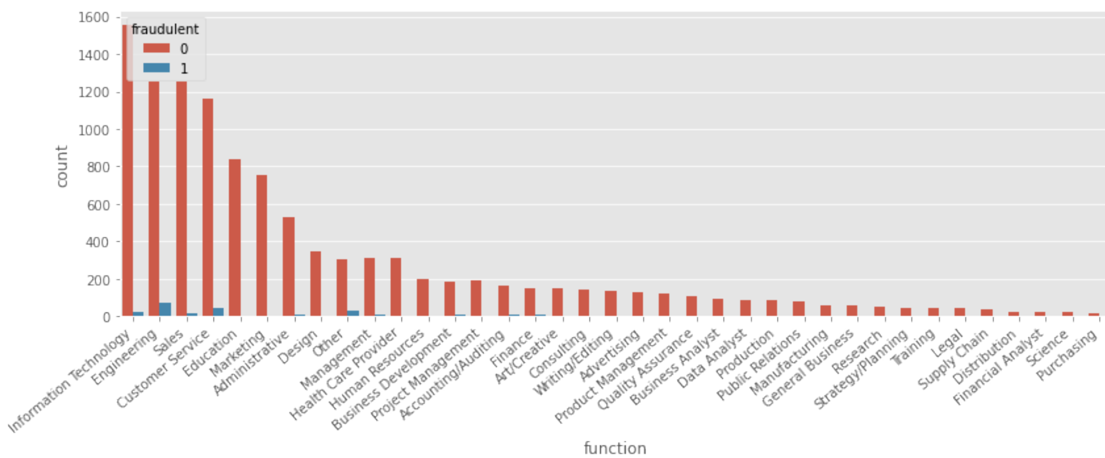
[29]:
```
plt.figure(figsize = (7, 5))
sns.countplot( x= 'employment_type' ,hue = 'fraudulent', data= df_clean, order
 = df_clean['employment_type'].value_counts().index)
plt.figure(figsize = (12, 5))
sns.countplot( x= 'required_experience' ,hue = 'fraudulent', data= df_clean,
 order = df_clean['required_experience'].value_counts().index)
plt.figure(figsize = (12, 5))
ax = sns.countplot( x= 'required_education' ,hue = 'fraudulent', data=
 df_clean, order = df_clean['required_education'].value_counts().index )
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.figure(figsize = (12, 5))
axa = sns.countplot( x= 'function' ,hue = 'fraudulent', data= df_clean, order =
 df_clean['function'].value_counts().index )
```

```
axa.set_xticklabels(axa.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
```









Natural Language Processing

From here starts a natutral language processing (NLP) part. It needs some libraries to download everytime when runtime is started.

[0]:
```
import nltk
nltk.download('popular')
```

[32]:
```
import spacy.cli
spacy.cli.download("en_core_web_lg")
```

```
 Download and installation successful
You can now load the model via spacy.load('en_core_web_lg')
```

[0]:
```python
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
from nltk.stem import WordNetLemmatizer
import string
import base64
import re
from collections import Counter
import spacy
spacy.load('en_core_web_sm')
nlp = spacy.load('en_core_web_lg')
punctuations = string.punctuation
from spacy.lang.en import English
parser = English()
```

[34]:
```python
df_clean['fraudulent'].value_counts()
```

[34]:
```
0    11023
1      249
Name: fraudulent, dtype: int64
```

[0]:
```python
def cleanup(docs, logging = False):
    texts = []
    counter = 1
    for doc in docs:
        if counter % 100 == 0 and logging:
            print ("Processed %d out of %d documents."%(counter, len(docs)))
        counter +=1
        doc = nlp(doc, disable = ['parser', 'ner'])
        tokens = [tok.lemma_.lower().strip() for tok in doc if tok.lemma_ !=␣
 ↪'-PRON-']
        tokens = [tok for tok in tokens if tok not in stop_words and tok not in␣
 ↪punctuations]
        tokens = ' '.join(tokens)
        texts.append(tokens)
    return pd.Series(texts)
```

This function is used for cleaning the feature named description. By using this, common words (Stopwords), pronouns and syambols are removed. So only words which are affecting the prediction can be used.

[0]:
```python
Fraud_1 = [text for text in df_clean[df_clean['fraudulent'] ==␣
 ↪1]['description']]
Fraud_0 = [te for te in df_clean[df_clean['fraudulent'] == 0]['description']]
```

[0]:
```python
Fraud_1_clean = cleanup(Fraud_1)
Fraud_1_clean = ' '.join(Fraud_1_clean).split()
```

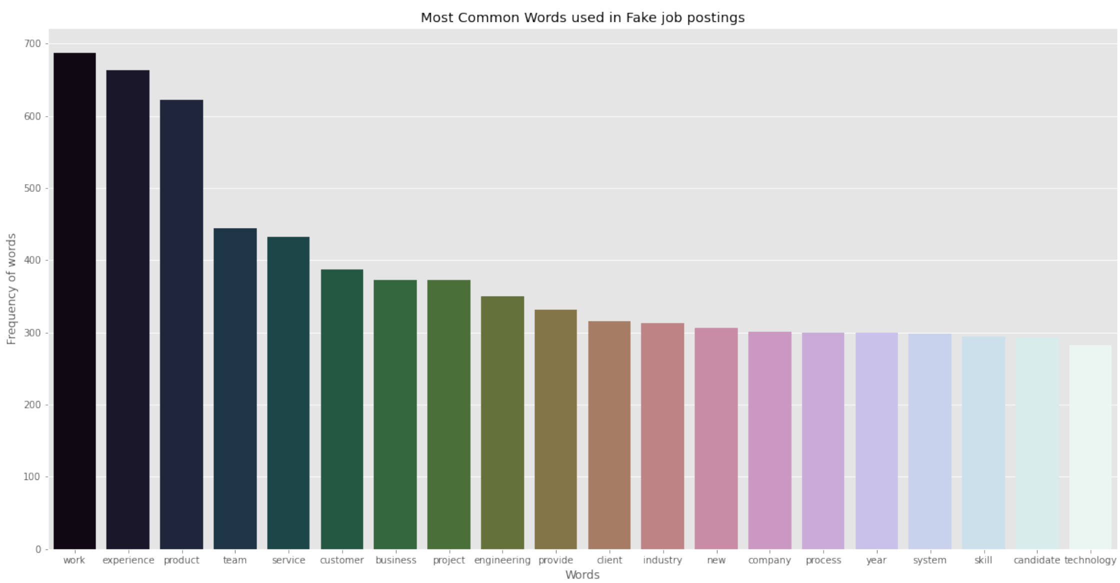
[0]:
```python
Fraud_0_clean = cleanup(Fraud_0)
Fraud_0_clean = ' '.join(Fraud_0_clean).split()
```

[39]:
```
print(len(Fraud_1_clean))
print(len(Fraud_0_clean))
```

```
64441
2770033
```

[0]:
```
Fraud_1_common_words = [word[0] for word in Counter(Fraud_1_clean).
  ↪most_common(20)]
Fraud_1_common_counts = [word[1] for word in Counter(Fraud_1_clean).
  ↪most_common(20)]
```

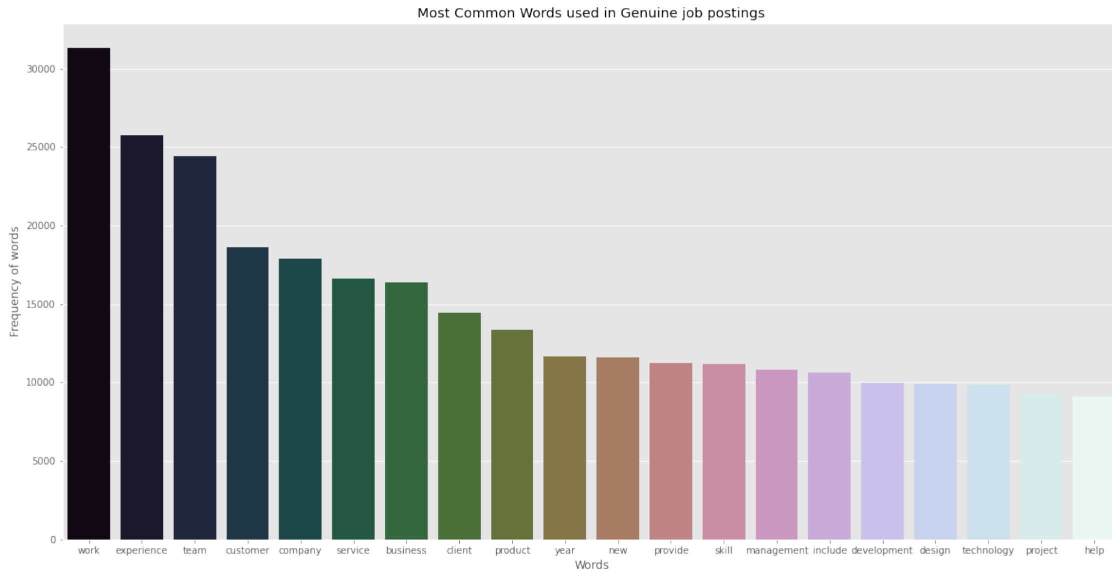
[166]:
```
fig = plt.figure(figsize = (20, 10))
pal = sns.color_palette("cubehelix", 20)
sns.barplot(x = Fraud_1_common_words, y = Fraud_1_common_counts, palette=pal)
plt.title('Most Common Words used in Fake job postings')
plt.ylabel("Frequency of words")
plt.xlabel("Words")
plt.show()
```



[0]:
```
Fraud_0_common_words = [word[0] for word in Counter(Fraud_0_clean).
  ↪most_common(20)]
Fraud_0_common_counts = [word[1] for word in Counter(Fraud_0_clean).
  ↪most_common(20)]
```

[168]:
```
fig = plt.figure(figsize = (20, 10))
pal = sns.color_palette("cubehelix", 20)
sns.barplot(x = Fraud_0_common_words, y = Fraud_0_common_counts, palette=pal)
plt.title('Most Common Words used in Genuine job postings')
plt.ylabel("Frequency of words")
```

```
plt.xlabel("Words")
plt.show()
```



The above graphs shows the words which are mostly occured in fake vs true job post.