Coursework 2 Online Fake Jobs Prediction

1. Read and Understand Datasets

Import Libraries

In [271]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
pd.set_option('display.max_columns', None)
```

In [272]:

```
fk_job = pd.read_csv('fake_job_postings.csv')
```

In [273]:

```
fk_job.head()
```

Out[273]:

	job_id	title	location	department	salary_range	company_profile	d
0	1	Marketing Intern	US, NY, New York	Marketing	NaN	We're Food52, and we've created a groundbreaki...	Fooc growing, Ja Aw
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Success	NaN	90 Seconds, the worlds Cloud Video Production ...	Organised - Vibrant - Aw
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	NaN	NaN	Valor Services provides Workforce Solutions th...	Our client Houston
3	4	Account Executive - Washington DC	US, DC, Washington	Sales	NaN	Our passion for improving quality of life thro...	THE COMP – Env Syste
4	5	Bill Review Manager	US, FL, Fort Worth	NaN	NaN	SpotSource Solutions LLC is a Global Human Cap...	J Itemizati ManagerLO

In [274]:

```
# To get the summary of rows and columns
fk_job.shape
```

Out[274]:

```
(17880, 18)
```

In [275]:

```
# To count the unique value exclude Nan values
fk_job.nunique()
```

Out[275]:

```
job_id                 17880
title                  11231
location                3105
department              1337
salary_range             874
company_profile         1709
description            14801
requirements           11968
benefits                6205
telecommuting              2
has_company_logo           2
has_questions              2
employment_type            5
required_experience        7
required_education        13
industry                 131
function                  37
fraudulent                 2
dtype: int64
```

In [276]:

```
# To check the missing value
fk_job.isna().sum() / len(fk_job)
```

Out[276]:

```
job_id                 0.000000
title                  0.000000
location               0.019351
department             0.645805
salary_range           0.839597
company_profile        0.185011
description            0.000056
requirements           0.150727
benefits               0.403244
telecommuting          0.000000
has_company_logo       0.000000
has_questions          0.000000
employment_type        0.194128
required_experience    0.394295
required_education     0.453300
industry               0.274217
function               0.361018
fraudulent             0.000000
dtype: float64
```

In [277]:

```
fk_job.info()
```

```
<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
```

In [278]:

```
# Create a new dataframe2
#df2 = df.copy()
```

In [279]:

```
# Summary counts of True and Fraud jobs
fk_job['fraudulent'].value_counts()
```

Out[279]:

```
0    17014
1      866
Name: fraudulent, dtype: int64
```

Exploratory Analysis

2.1 Data Transformation

In [280]:

```
# Features Selections by delist some columns with high missing value
fk_job.drop(['department', 'benefits', 'salary_range'], axis = 1, inplace = True)
```

In [281]:

```
# Review new dataframe
fk_job.head()
```

Out[281]:

	job_id	title	location	company_profile	description	require
0	1	Marketing Intern	US, NY, New York	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experience with management sys
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome!Do you...	What we expe you:Y respon
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Impleme commissioni commissio
3	4	Account Executive - Washington DC	US, DC, Washington	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Bac or Master's
4	5	Bill Review Manager	US, FL, Fort Worth	SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATIO license in the S

In [282]:

```
# To check the unique values exclude the Nan values
#fkjob.nunique()
fk_job.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17880 entries, 0 to 17879
Data columns (total 15 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   company_profile      14572 non-null  object
 4   description          17879 non-null  object
 5   requirements         15185 non-null  object
 6   telecommuting        17880 non-null  int64 
 7   has_company_logo     17880 non-null  int64 
 8   has_questions        17880 non-null  int64 
 9   employment_type      14409 non-null  object
 10  required_experience  10830 non-null  object
 11  required_education   9775 non-null   object
 12  industry             12977 non-null  object
 13  function             11425 non-null  object
 14  fraudulent           17880 non-null  int64 
dtypes: int64(5), object(10)
memory usage: 2.0+ MB
```

In [121]:

```
# Sort columns of title in ascending order
#df2 = df2.sort_values('title').reset_index(drop = True)
#df2.head()
```

In [283]:

```
# To check the misiing value of df2
fk_job.isna().sum()
```

Out[283]:

```
job_id                    0
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
```

In [284]:

```
# To fill the missing value with 0 (axis=0)
fk_job['employment_type'] = fk_job['employment_type'].bfill(axis=0)
fk_job['required_experience'] = fk_job['required_experience'].bfill(axis=0)
fk_job['required_education'] = fk_job['required_education'].bfill(axis=0)
fk_job['industry'] = fk_job['industry'].bfill(axis=0)
fk_job['function'] = fk_job['function'].bfill(axis=0)
```

In [15]:

```
# Create another new dataframe df3
#f3 = df2.copy()
```

In [16]:

```
# Detect existing non missing value
#df3 = df3[df3['description'].notna()]
```

In [127]:

```
# Check the count of df3
#df3.count()
```

In [285]:

```
# List with the columns to check the length of the text
feature_l = ['company_profile', 'description', 'requirements']

# For loop to treat the missing values in the columns of the feature_lst.
for col in feature_l:
    # If the job post is real, change the missing values to "none"
    fk_job.loc[(fkjob[col].isnull()) & (fk_job["fraudulent"] == 0), col] = "none"

    # If the job post is fake, change the missing values to "missing"
    fk_job.loc[(fk_job[col].isnull()) & (fk_job["fraudulent"] == 1), col] = "missing"
```

In [286]:

```
# For loop to create new columns with the lengths of the ones in the feature_lst
for num,col in enumerate(feature_l):
    fk_job[str(num)] = fk_job[col].apply(len)
```

In [287]:

```
# Rename the new columns created above
fk_job = fk_job.rename({"0": "profile_length", "1": "description_length", "2": "require
ments_length"}, axis=1)
```

In [288]:

```
# Check the details of missing value
fk_job.isna().sum()
```

Out[288]:

```
job_id                   0
title                    0
location               346
company_profile          0
description              0
requirements             0
telecommuting            0
has_company_logo         0
has_questions            0
employment_type          0
required_experience      0
required_education       1
industry                 0
function                 0
fraudulent               0
profile_length           0
description_length       0
requirements_length      0
dtype: int64
```

In [289]:

```
# Summary counts of True and Fraud jobs
fk_job['fraudulent'].value_counts()
```

Out[289]:

```
0    17014
1      866
Name: fraudulent, dtype: int64
```

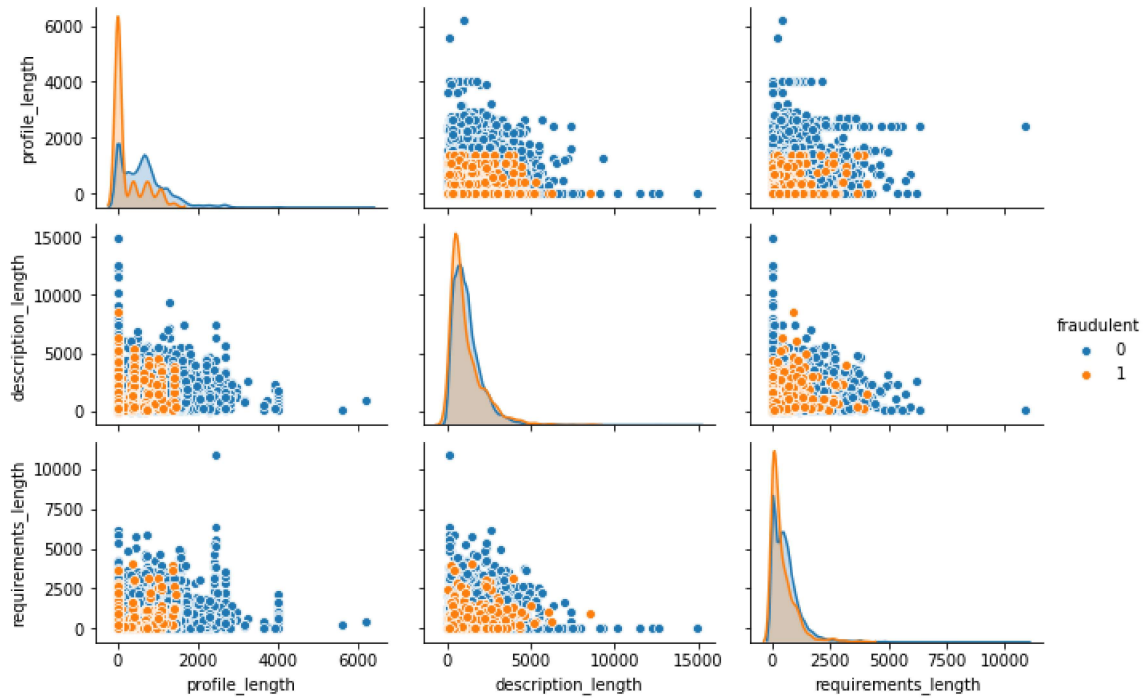
2.2 Visualizations

In [140]:

```
# Drop the index with missing value label
#df3 = df3.dropna(axis = 0, how ='any')

# Check df3 after drop missing value label
#df3.count()

# Check columns and rows of new dataframe
#df3.shape

# Delist duplicates index
#df3 = df3.drop_duplicates(keep = 'first')

# Check rows and columns of df3
#df3.shape

# Duplicates a new dataframe df4
#df4 = df3.copy()
```

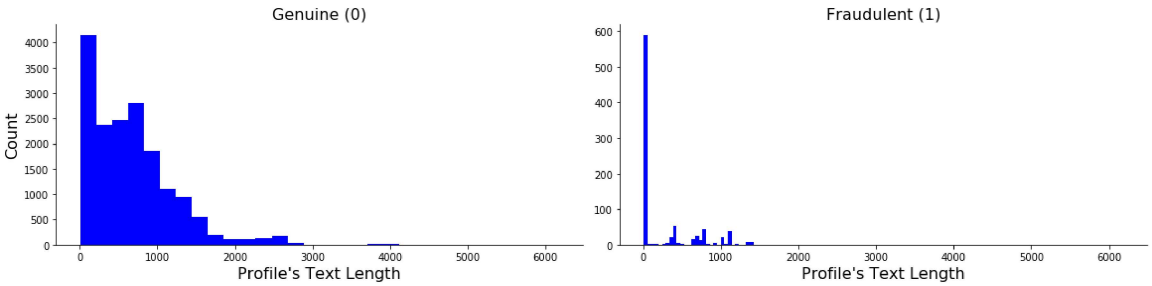
In [291]:

```
sns.pairplot(data=fk_job[["fraudulent", "profile_length", "description_length", "requir
ements_length"]],
             hue="fraudulent", height=2, aspect=1.5);
```



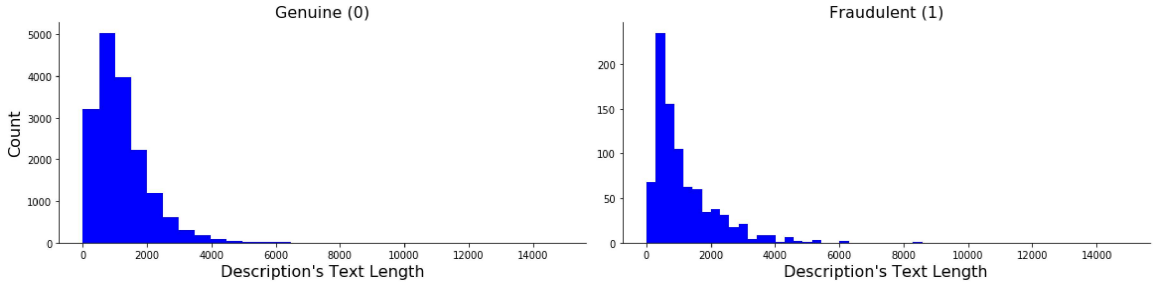
In [292]:

```
prof_grid = sns.FacetGrid(fk_job, col="fraudulent", aspect=2, height=4, sharey=False)
prof_grid = prof_grid.map(plt.hist, "profile_length", bins=30, color = 'b')

# Flatten the axes. Create an iterator
axes = prof_grid.axes.flatten()

# Title
axes[0].set_title("Genuine (0)", fontsize=16)
axes[1].set_title("Fraudulent (1)", fontsize=16)

# Labels
axes[0].set_ylabel("Count", fontsize=16)

for ax in axes:
    ax.set_xlabel("Profile's Text Length", fontsize=16)
```



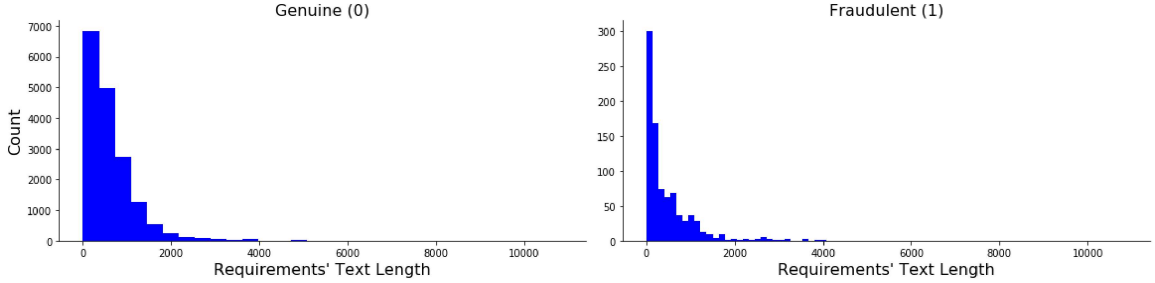
In [293]:

```
descpt_grid = sns.FacetGrid(fk_job, col="fraudulent", aspect=2, height=4, sharey=False)
descpt_grid = descpt_grid.map(plt.hist, "description_length", bins=30, color = 'b')

# Flatten the axes. Create an iterator
axes = descpt_grid.axes.flatten()

# Title
axes[0].set_title("Genuine (0)", fontsize=16)
axes[1].set_title("Fraudulent (1)", fontsize=16)

# Labels
axes[0].set_ylabel("Count", fontsize=16)
for ax in axes:
    ax.set_xlabel("Description's Text Length", fontsize=16)
```



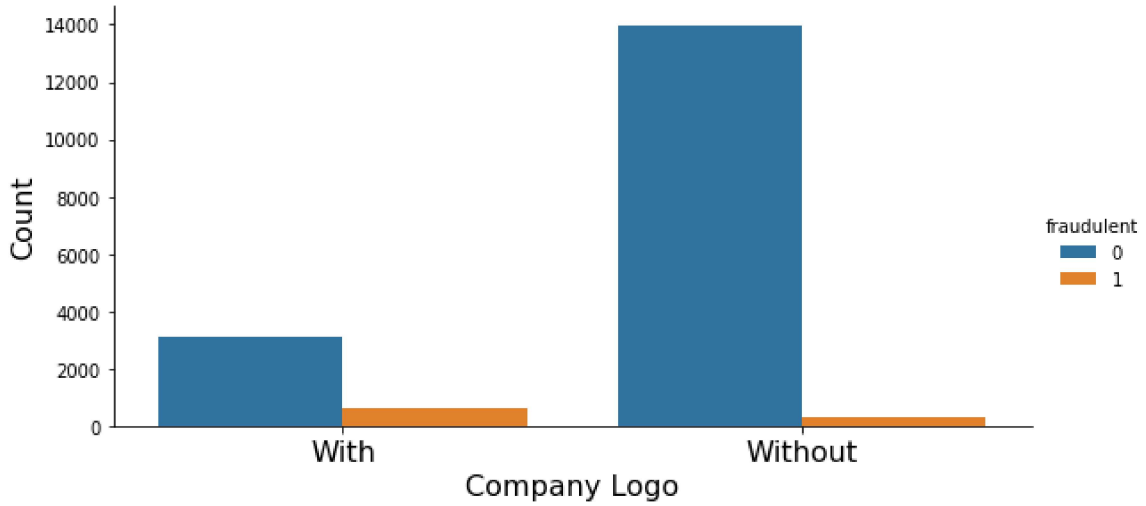
In [294]:

```
requirt_grid = sns.FacetGrid(fk_job, col="fraudulent", aspect=2, height=4, sharey=False
)
requirts_grid = requirt_grid.map(plt.hist, "requirements_length", bins=30, color = 'b')

# Another option. Makes less obviuos which axes is to be labelled
#requirements_grid.set_axis_labels("Requirement Length", "Count")

# Flatten the axes. Create an iterator
axes = requirt_grid.axes.flatten()

# Title
axes[0].set_title("Genuine (0)", fontsize=16)
axes[1].set_title("Fraudulent (1)", fontsize=16)

# Labels
axes[0].set_ylabel("Count", fontsize=16)
for ax in axes:
    ax.set_xlabel("Requirements' Text Length", fontsize=16)
```



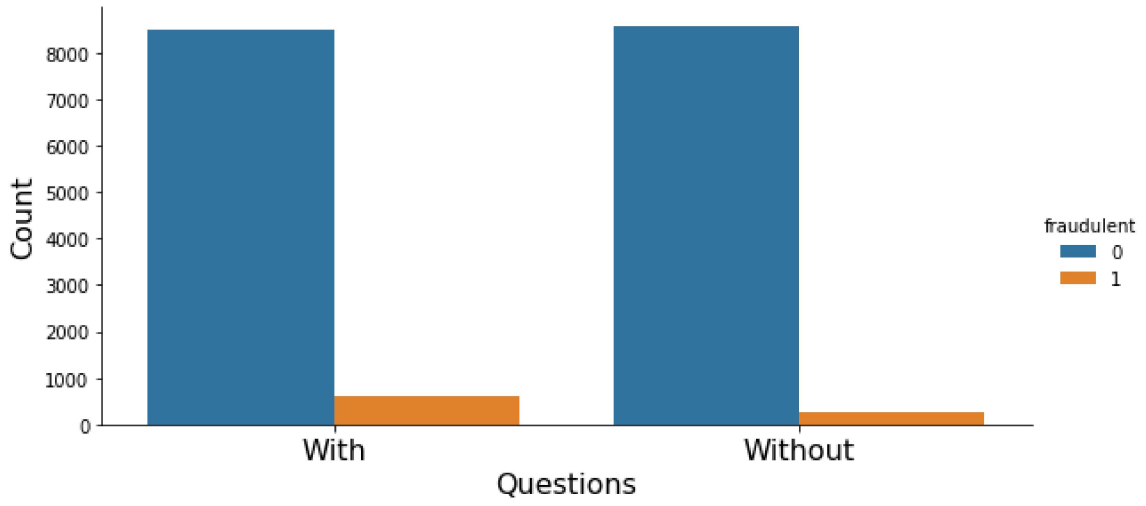
In [295]:

```
sns.catplot(x="has_company_logo", hue="fraudulent", data=fk_job, kind="count", aspect=2
, height=4);

plt.xlabel("Company Logo", fontsize=16)
plt.xticks([0, 1], ("With", "Without"), fontsize=16)
plt.ylabel("Count", fontsize=16);
```



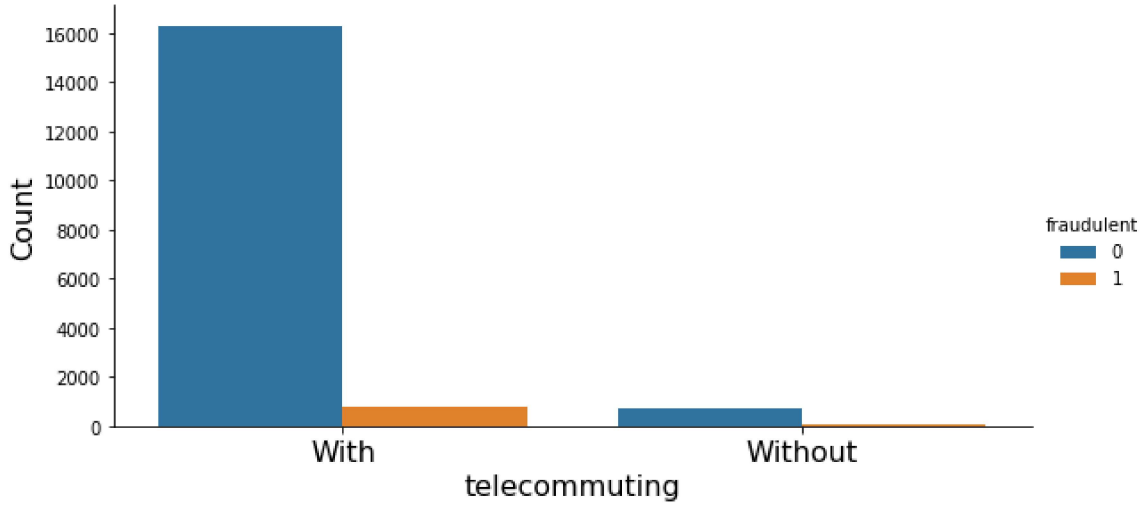
In [296]:

```
sns.catplot(x="has_questions", hue="fraudulent", data=fk_job, kind="count", aspect=2, h
eight=4);

plt.xlabel("Questions", fontsize=16)
plt.xticks([0, 1], ("With", "Without"), fontsize=16)
plt.ylabel("Count", fontsize=16);
```



In [297]:

```
sns.catplot(x="telecommuting", hue="fraudulent", data=fk_job, kind="count", aspect=2, h
eight=4);

plt.xlabel("telecommuting", fontsize=16)
plt.xticks([0, 1], ("With", "Without"), fontsize=16)
plt.ylabel("Count", fontsize=16);
```



In [298]:

```
# Consolidates some columns/ features
fk_job['description'] = fk_job['description'] + '' + fk_job['requirements'] + '' + fk_j
ob['company_profile']
fk_job.drop(['company_profile', 'requirements'], axis = 1, inplace = True)
```

In [299]:

```
# Split country code column into location and city
fk_job['country_code'] = fk_job['location'].str.split(',', expand=True)[0]
fk_job['city'] = fk_job['location'].str.split(',', expand = True) [2]
```

In [300]:

```
fk_job.head()
```

Out[300]:

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

In [301]:

```
#Check occurance of nan (np.nan)
#df4.loc[df4['city'] == '', 'city'] = np.nan
```

In [302]:

```
#df4.head()
```

In [303]:

```
fk_job.isnull().sum()
```

Out[303]:

```
job_id                   0
title                    0
location               346
description              0
telecommuting            0
has_company_logo         0
has_questions            0
employment_type          0
required_experience      0
required_education       1
industry                 0
function                 0
fraudulent               0
profile_length           0
description_length       0
requirements_length      0
country_code           346
city                   440
dtype: int64
```

In [304]:

```
# Drp rows and columns with null values
#df4.dropna(inplace = True)
```

In [305]:

```
fk_job.shape
```

Out[305]:

```
(17880, 18)
```

In [306]:

```
# install wordcloud into jupyter
#!pip install wordcloud
```

In [307]:

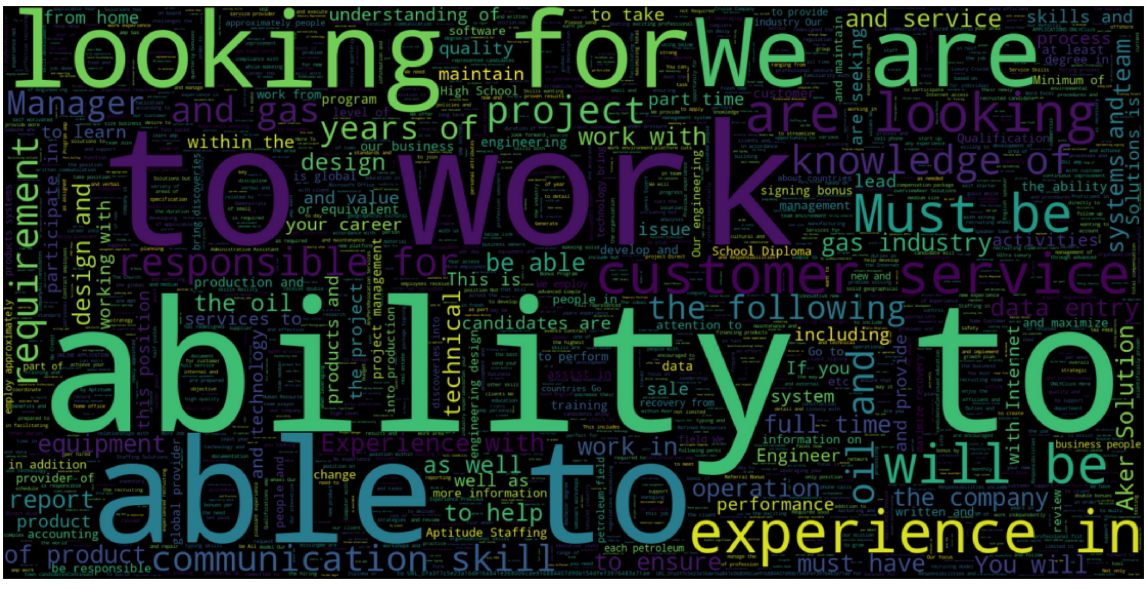
```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [308]:

```
plt.figure(figsize = (20,20))
stopwords = set(STOPWORDS)
wc = WordCloud(background_color = 'white', stopwords = stopwords, width =1600, height =
800, max_words = 3000).generate(''.join(fk_job[fk_job.fraudulent == 0]['description']))
plt.axis('off')
plt.imshow(wc, interpolation = 'bilinear')
plt.savefig('Genuine_cloud.jpeg')
```

In [309]:

```
plt.figure(figsize = (20,20))
stopwords = set(STOPWORDS)
wc = WordCloud(background_color = 'black', stopwords=stopwords, width = 1600 , height =
800 , max_words = 3000).generate(''.join(fk_job[fk_job.fraudulent == 1]['description'
]))
plt.axis("off")
plt.imshow(wc , interpolation = 'bilinear')
plt.savefig('fraud_cloud.jpeg')
```



In [310]:

```
# create new df
#df_clean = df4.copy()
#df_clean.head()
#df_clean.isnull().sum()
# To delete column location as have been dupilcate columns of country_code and city
#del df_clean['location']
```

3. Deploy of Machine Learning Model

In [311]:

```
#conda install -c conda-forge spacy
#import sys
#!{sys.executable} -m pip install -U spacy
```

In [312]:

```
#import nltk
#nltk.download('stopwords')
```

In [313]:

```
import base64
import string
import re

from collections import Counter
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
nlp = spacy.load('en_core_web_lg')
stop_words = set(stopwords.words("english"))
from nltk.stem import WordNetLemmatizer

import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.base import TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.stop_words import ENGLISH_STOP_WORDS
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

#import spacy
#spacy.load('en_core_web_sm')
#from spacy.lang.en import English
#parser = English()
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
```

In [314]:

```
import spacy
from spacy.lang.en import English
nlp = spacy.load('en_core_web_lg')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

In [50]:

```
#!python -m spacy download en_core_web_lg
```

In [51]:

```
#!python -m spacy download en_core_web_sm
```

In [315]:

```
punctuations = string.punctuation
```

In [316]:

```
def cleanup_text(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 punctua
tions]
        tokens = ' '.join(tokens)
        texts.append(tokens)
    return pd.Series(texts)
```

In [317]:

```
fake_1 = [text for text in fk_job[fk_job['fraudulent'] == 1]['description']]
```

In [318]:

```
fake_1[10]
```

Out[318]:

'Corporate overviewAker Solutions is a global provider of products, system
s and services to the oil and gas industry. Our engineering, design and te
chnology bring discoveries into production and maximize recovery from each
petroleum field. We employ approximately 28,000 people in about 30 countri
es. Go to #URL_0fa3f7c5e23a16de16a841e368006cae916884407d90b154dfef3976483
a71ae# for more information on our business, people and values.We are look
ing for individuals who are prepared to take a position. Not only a positi
on within Aker Solutions, but also a position on the exciting challenges t
he global oil and gas industry faces now and in the future.We are looking
for a Lead Mechanical Engineer to join our team in Houston, Texas.The Lead
Mechanical Engineer will be responsible for providing expertise and techni
cal leadership to the organization.Responsibilities and tasks• Performs me
chanical calculations and technical analysis on various custom components
and reviews mechanical design of equipment to ensure that specifications a
re met;• Prepares and presents complex technical reports, equipment data s
heets, MRQ’s, TBE’s and MRP’s, and makes recommendations on critical engin
eering issues;• Work with certifying agencies for product development and
follows through with ABSA registrations;• Leads and reviews project design
decisions, budgets, and scheduling;• Identifies solutions to achieve compa
ny objectives and ensure that the team is aligned.• Interfaces directly wi
th the customer and participates in preparing bids and proposals;• Ensure
processes are followed correctly and continuously identifies opportunities
to improve efficiencies;• Ensure team members are kept current on procedur
e QMS changes;• Provides leadership, technical guidance and mentor-ship to
other engineers.Qualifications & personal attributes• Mechanical Engin
eering Degree (or equivalent) is required.• 5-10 years related experience
within an EPC, Oil & Gas, Fabrication shop and/or Engineering environm
ent is required.• Registration with APEGGA (or eligibility to be a member)
is required.• Must have experience with different types of mechanical equi
pment including Pressure Vessels, Pumps, Heat Exchangers.• Familiarity wit
h the industry codes relevant to the above equipment, specifically relevan
t ASME and API 610.• Ability to effectively present information and respon
d to questions from managers, employees, customers and the general public•
Proficiency with Microsoft Office applications• Excellent time management/
prioritization skills with the ability to work effectively with minimal su
pervision and manage multiple, conflicting tasks/projectsAker Solutions is
a global provider of products, systems and services to the oil and gas ind
ustry. Our engineering, design and technology bring discoveries into produ
ction and maximize recovery from each petroleum field. We employ approxima
tely 28,000 people in about 30 countries. Go to #URL_0fa3f7c5e23a16de16a84
1e368006cae916884407d90b154dfef3976483a71ae# for more information on our b
usiness, people and values.'

In [319]:

```
fake_0 = [text for text in fk_job[fk_job['fraudulent'] == 0]['description']]
```

In [320]:

```
fake_1_clean = cleanup_text(fake_1)
fake_1_clean = ' '.join(fake_1_clean).split()
```

In [321]:

```
len(fake_1_clean)
```

Out[321]:

148557

In [322]:

```
fake_0_clean = cleanup_text(fake_0)
fake_0_clean = ' '.join(fake_0_clean).split()
```

In [323]:

```
len(fraud_0_clean)
```

Out[323]:

2993981

In [324]:

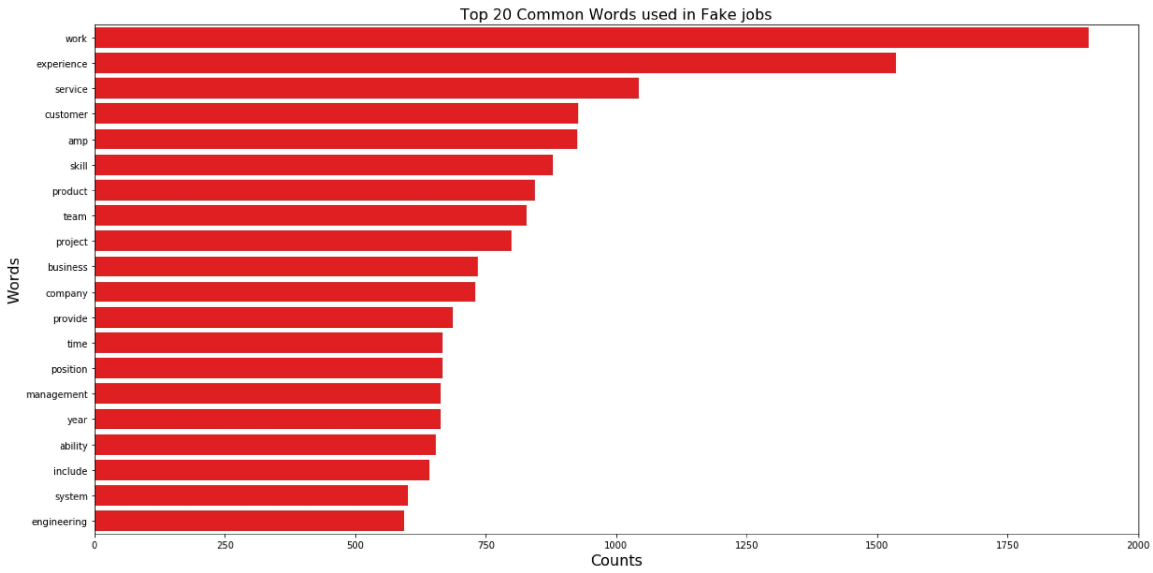
```
fake_1_counts = Counter(fake_1_clean)
fake_0_counts = Counter(fake_0_clean)
```

In [325]:

```
fake_1_common_words = [word[0] for word in fake_1_counts.most_common(20)]
fake_1_common_counts = [word[1] for word in fake_1_counts.most_common(20)]
```

In [331]:

```
fig = plt.figure(figsize = (20, 10))
pal = sns.color_palette("cubehelix", 20)
sns.barplot(y = fake_1_common_words, x = fake_1_common_counts, color = 'r')
plt.title('Top 20 Common Words used in Fake jobs', fontsize=16)
plt.xlabel("Counts", fontsize=16)
plt.ylabel("Words", fontsize=16)
plt.show()
```

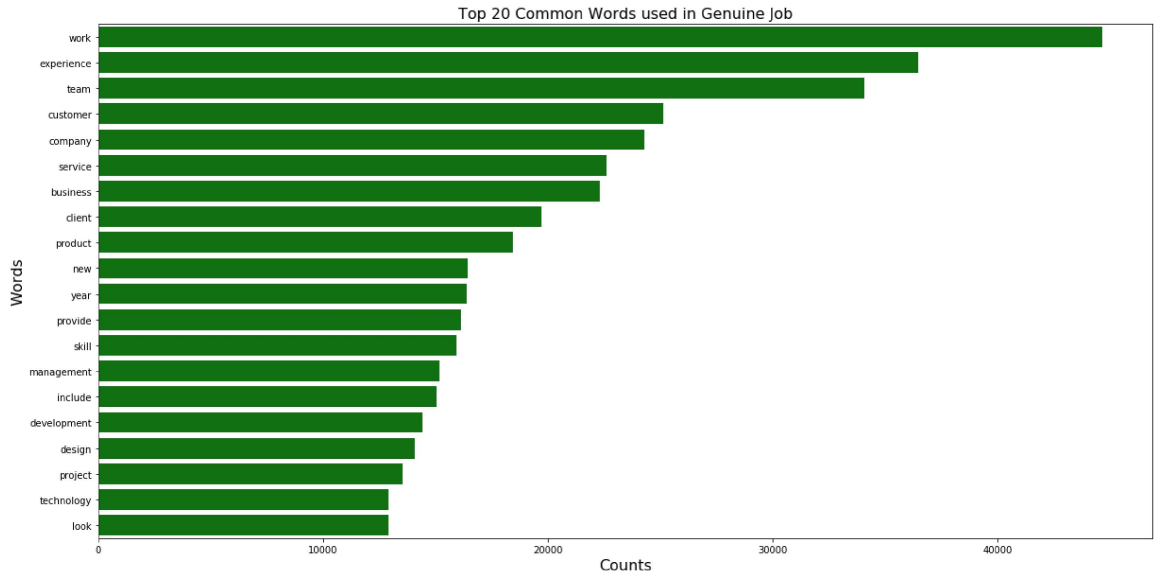


In [329]:

```
fake_0_common_words = [word[0] for word in fake_0_counts.most_common(20)]
fake_0_common_counts = [word[1] for word in fake_0_counts.most_common(20)]
```

In [330]:

```
fig = plt.figure(figsize = (20, 10))
pal = sns.color_palette("cubehelix", 20)
sns.barplot(y = fake_0_common_words, x = fake_0_common_counts, color = 'g')
plt.title('Top 20 Common Words used in Genuine Job', fontsize=16)
plt.xlabel("Counts", fontsize=16)
plt.ylabel("Words", fontsize=16)
plt.show()
```

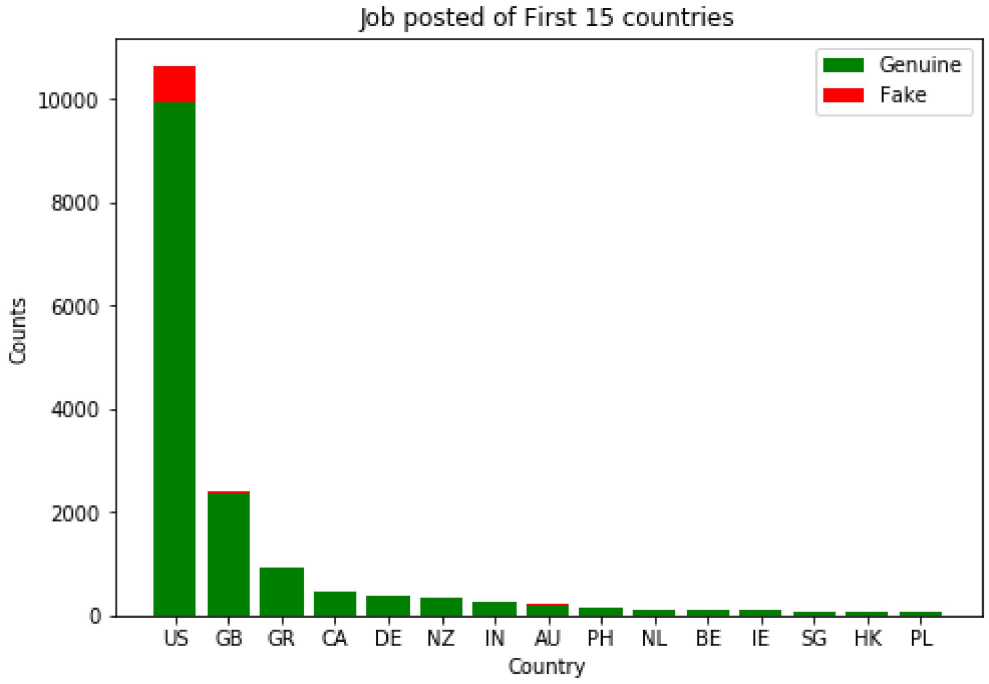


In [332]:

```
ccount = fk_job.groupby(['country_code', 'fraudulent']).size().unstack('fraudulent', fi
ll_value=0)
ccount = ccount.sort_values(by=[0, 1], ascending=False)
ccount_15 = ccount[:15] # Take top 15 countries
fig = plt.figure()
axi = fig.add_axes([0, 0, 1, 1])
axi.bar(ccount_15.index, ccount_15[0], color = 'g')
axi.bar(ccount_15.index, ccount_15[1], bottom = ccount_15[0], color = 'r')
axi.set_ylabel('Counts')
axi.set_xlabel('Country')
axi.set_title('Job posted of First 15 countries')
axi.legend(labels=['Genuine', 'Fake'])
```

Out[332]:

```
<matplotlib.legend.Legend at 0x25c5964c448>
```



Final Data Cleaning

In [73]:

```
#!pip install keras
```

In [74]:

```
#!pip install tensorflow
```

In [333]:

```
import tensorflow as tf
from tensorflow import keras
from keras.wrappers.scikit_learn import KerasClassifier
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

In [334]:

```
stoplist = set(stopwords.words('english')+ list(ENGLISH_STOP_WORDS))
symbol = ' '.join(string.punctuation).split(' ')
```

In [335]:

```
def tokenizetext(sample):
    text = sample.strip().replace("\n", " ").replace("\r", " ")
    text = text.lower()
    tokens = parse(text)
    lemmas = []
    for tok in tokens:
        lemmas.append(tok.lemma_.lower().strip() if tok.lemma_ != "-PRON-" else tok.low
er_)
    tokens = lemmas
    tokens = [tok for tok in tokens if tok not in stoplist]
    tokens = [tok for tok in tokens if tok not in symbol]
    return tokens
```

In [336]:

```
vectorizer = CountVectorizer(tokenizer = tokenizetext, ngram_range = (1,2), min_df = 0.
05)
```

In [337]:

```
fk_job.info()
```

```
<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   description          17880 non-null  object
 4   telecommuting        17880 non-null  int64 
 5   has_company_logo     17880 non-null  int64 
 6   has_questions        17880 non-null  int64 
 7   employment_type      17880 non-null  object
 8   required_experience  17880 non-null  object
 9   required_education   17879 non-null  object
 10  industry             17880 non-null  object
 11  function             17880 non-null  object
 12  fraudulent           17880 non-null  int64 
 13  profile_length       17880 non-null  int64 
 14  description_length   17880 non-null  int64 
 15  requirements_length  17880 non-null  int64 
 16  country_code         17534 non-null  object
 17  city                 17440 non-null  object
dtypes: int64(8), object(10)
memory usage: 2.5+ MB
```

In [338]:

```
text_columns = ['title', 'location', 'description', 'employment_type','required_experie
nce','required_education','industry','function', 'country_code', 'city']
```

In [339]:

```
#text_columns
```

In [340]:

```
#vectorizer_features = vectorizer.fit_transform(text_columns)
```

In [341]:

```
max_length = 100
vocab_size = 1500
embedding_dim = 32
text = {}
text['descriptions'] = fk_job['description'].to_numpy()
text['labels'] = fk_job['fraudulent'].to_numpy()
tokenizer = Tokenizer(num_words = vocab_size)
tokenizer.fit_on_texts(text['descriptions'])
sequences = tokenizer.texts_to_sequences(text['descriptions'])
padded_sequences = pad_sequences(sequences, maxlen = max_length, padding = 'post')
```

In [342]:

```
x_train, x_test, y_train, y_test = train_test_split(padded_sequences, text['labels'], t
est_size = 0.25, random_state=0)
```

Import Libraries for Machine Learning

In [343]:

```
Model_Score = []
Model_Name = []
```

In [344]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ro
c_auc_score
```

1.GaussianNB

In [345]:

```
gnb = GaussianNB()

gnb.fit(x_train, y_train)
y_predict = gnb.predict(x_test)
accuracy_score(y_test, y_predict)
```

Out[345]:

```
0.8451901565995525
```

In [346]:

```
gnb_pred = gnb.predict(x_test)
roc_auc_score(y_test, gnb_pred)
print ('roc_auc_score : ', roc_auc_score(y_test, gnb_pred))
```

```
roc_auc_score :  0.6914470628346328
```

In [347]:

```
print ('Gnb:', classification_report(y_test, gnb_pred))
```

```
Gnb:               precision    recall  f1-score   support

           0       0.98      0.86      0.91      4273
           1       0.15      0.52      0.23       197

    accuracy                           0.85      4470
   macro avg       0.56      0.69      0.57      4470
weighted avg       0.94      0.85      0.88      4470
```

In [348]:

```
Model_Name.append('GaussianNB')
Model_Score.append(round(roc_auc_score(y_test, gnb_pred), 4))
```

2. Logistic Regression

In [349]:

```
log_reg = LogisticRegression()
c_values = [0.001, 0.01, 0.1, 1, 10, 100]
penalty_options = ['8', '10', 'l2']

param_grid = dict(C = c_values, penalty = penalty_options)
grid_log = GridSearchCV(log_reg, param_grid = param_grid, cv = 10, scoring = 'roc_auc',
n_jobs = 1, verbose=1)
```

In [350]:

```
grid_log.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 18 candidates, totalling 180 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 8.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 10.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 8.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 10.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 8.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 10.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 8.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 10.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
```

```
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 8.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 10.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 8.

  FitFailedWarning)
C:\Users\meita\anaconda3\lib\site-packages\sklearn\model_selection\_valida
tion.py:536: FitFailedWarning: Estimator fit failed. The score on this tra
in-test partition for these parameters will be set to nan. Details:
ValueError: Logistic Regression supports only penalties in ['l1', 'l2', 'e
lasticnet', 'none'], got 10.

  FitFailedWarning)
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:   11.8s finished
```

Out[350]:

```
GridSearchCV(cv=10, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=F
alse,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=No
ne,
                                          max_iter=100, multi_class='aut
o',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver='lbfg
s',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=1,
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
                         'penalty': ['8', '10', 'l2']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring='roc_auc', verbose=1)
```

In [351]:

```
grid_log.best_score_
grid_log.best_params_
print ('grid_log.best_score : ', grid_log.best_score_)
print ('grid_log.best_params : ', grid_log.best_params_)
```

```
grid_log.best_score :  0.7325057300379011
grid_log.best_params :  {'C': 10, 'penalty': 'l2'}
```

In [352]:

```
log_reg_pred = grid_log.predict(x_test)
roc_auc_score(y_test, log_reg_pred)
print ('roc_auc_score : ', roc_auc_score(y_test, log_reg_pred))
```

```
roc_auc_score :  0.5151114125883098
```

In [353]:

```
print ('Log:', classification_report(y_test, log_reg_pred))
```

```
Log:               precision    recall  f1-score   support

           0       0.96      1.00      0.98      4273
           1       0.86      0.03      0.06       197

    accuracy                           0.96      4470
   macro avg       0.91      0.52      0.52      4470
weighted avg       0.95      0.96      0.94      4470
```

In [354]:

```
Model_Name.append('Logistic_Regression')
Model_Score.append(round(roc_auc_score(y_test, log_reg_pred), 4))
```

3. Random Forest Classifier

In [355]:

```
random_fc = RandomForestClassifier()
n_estimators_range = [50, 100, 150, 200]
param_grid_rf = dict(n_estimators = n_estimators_range)
grid_rfc = GridSearchCV(random_fc, param_grid_rf, cv = 10, scoring = 'roc_auc', n_jobs
= 1, verbose = 1)
```

In [356]:

```
grid_rfc.fit(x_train, y_train)
grid_rfc.best_score_
grid_rfc.best_params_
print ('grid_rfc.best_score : ', grid_rfc.best_score_)
print ('grid_rfc.best_params : ', grid_rfc.best_params_)
```

```
Fitting 10 folds for each of 4 candidates, totalling 40 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:  6.2min finished

grid_rfc.best_score :  0.941322337319578
grid_rfc.best_params :  {'n_estimators': 200}
```

In [357]:

```
rfc_pred = grid_rfc.predict(x_test)
roc_auc_score(y_test, rfc_pred)
print ('roc_auc_score : ', roc_auc_score(y_test, rfc_pred))
```

```
roc_auc_score :  0.7817258883248731
```

In [358]:

```
print ('Rfc:', classification_report(y_test, rfc_pred))
```

```
Rfc:               precision    recall  f1-score   support

           0       0.98      1.00      0.99      4273
           1       1.00      0.56      0.72       197

    accuracy                           0.98      4470
   macro avg       0.99      0.78      0.86      4470
weighted avg       0.98      0.98      0.98      4470
```

In [359]:

```
Model_Name.append('Random_Forest')
Model_Score.append(round(roc_auc_score(y_test, rfc_pred), 4))
```

4. KNN

In [360]:

```
knn = KNeighborsClassifier()
k_range = list(np.arange(5, 30, 5))
param_grid_knn = dict(n_neighbors = k_range)
print (param_grid_knn)
```

```
{'n_neighbors': [5, 10, 15, 20, 25]}
```

In [361]:

```
grid_knn = GridSearchCV(knn, param_grid_knn, cv = 10, scoring = 'roc_auc', n_jobs = 1,
verbose = 1)
```

In [362]:

```
grid_knn.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 5 candidates, totalling 50 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  2.8min finished
```

Out[362]:

```
GridSearchCV(cv=10, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=3
0,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=Non
e,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=1,
             param_grid={'n_neighbors': [5, 10, 15, 20, 25]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring='roc_auc', verbose=1)
```

In [363]:

```
grid_knn.best_score_
grid_knn.best_params_
print ('grid_knn.best_score : ', grid_knn.best_score_)
print ('grid_knn.best_params : ', grid_knn.best_params_)
```

```
grid_knn.best_score :  0.8435524737118868
grid_knn.best_params :  {'n_neighbors': 15}
```

In [ ]:

```

```

In [364]:

```
knn_pred = grid_knn.predict(x_test)
print ('roc_auc_score : ', roc_auc_score(y_test, knn_pred))
```

```
roc_auc_score :  0.6597452306478764
```

In [365]:

```
print ('Knn:', classification_report(y_test, knn_pred))
```

```
Knn:               precision    recall  f1-score   support

           0       0.97      0.99      0.98      4273
           1       0.74      0.32      0.45       197

    accuracy                           0.97      4470
   macro avg       0.85      0.66      0.72      4470
weighted avg       0.96      0.97      0.96      4470
```

In [366]:

```
Model_Name.append('KNN')
Model_Score.append(round(roc_auc_score(y_test, knn_pred), 4))
```

5. Sklearn's MLP CIssifier (solver = 'lbfgs')

In [367]:

```
mlp = MLPClassifier(solver = 'sgd', activation = 'relu', hidden_layer_sizes = (100, 50,
30), max_iter = 1000)
```

In [369]:

```
mlp.fit(x_train, y_train)
```

Out[369]:

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=
0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100, 50, 30), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='sgd',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

In [370]:

```
mlp_pred = mlp.predict(x_test)
print ('roc_auc_score : ',roc_auc_score(y_test, mlp_pred))
print ('MLP', classification_report(y_test, mlp_pred))
```

```
roc_auc_score :  0.7260219700848558
MLP               precision    recall  f1-score   support

           0       0.98      0.97      0.97      4273
           1       0.42      0.48      0.45       197

    accuracy                           0.95      4470
   macro avg       0.70      0.73      0.71      4470
weighted avg       0.95      0.95      0.95      4470
```

In [371]:

```
Model_Name.append('MLP-NN (sgd)')
Model_Score.append(round(roc_auc_score(y_test, mlp_pred), 4))
```

In [ ]:

```

```

4. SVC

In [99]:

```
#svc = SVC()
#kernel = ['linear', 'rbf']
#param_grid_knn = dict(kernel = kernel)
#grid_svc = GridSearchCV(svc, param_grid_knn, cv = 10, scoring = 'roc_auc', n_jobs = -
1, verbose = 2)
```

In [ ]:

```
#grid_svc.fit(x_train, y_train)
```

In [ ]:

```
#grid_svc.best_score_
#grid_svc.best_params_
#print ('grid_svc.best_score : ', grid_svc.best_score_)
#print ('grid_svc.best_params : ', grid_svc.best_params_)
```

In [ ]:

```
#svc_pred = grid_svc.predict(x_test)
#roc_auc_score(y_test, svc_pred)
#print ('roc_auc_score : ', roc_auc_score(y_test, svc_pred))
#print ('SVC:', classification_report(y_test, svc_pred))
```

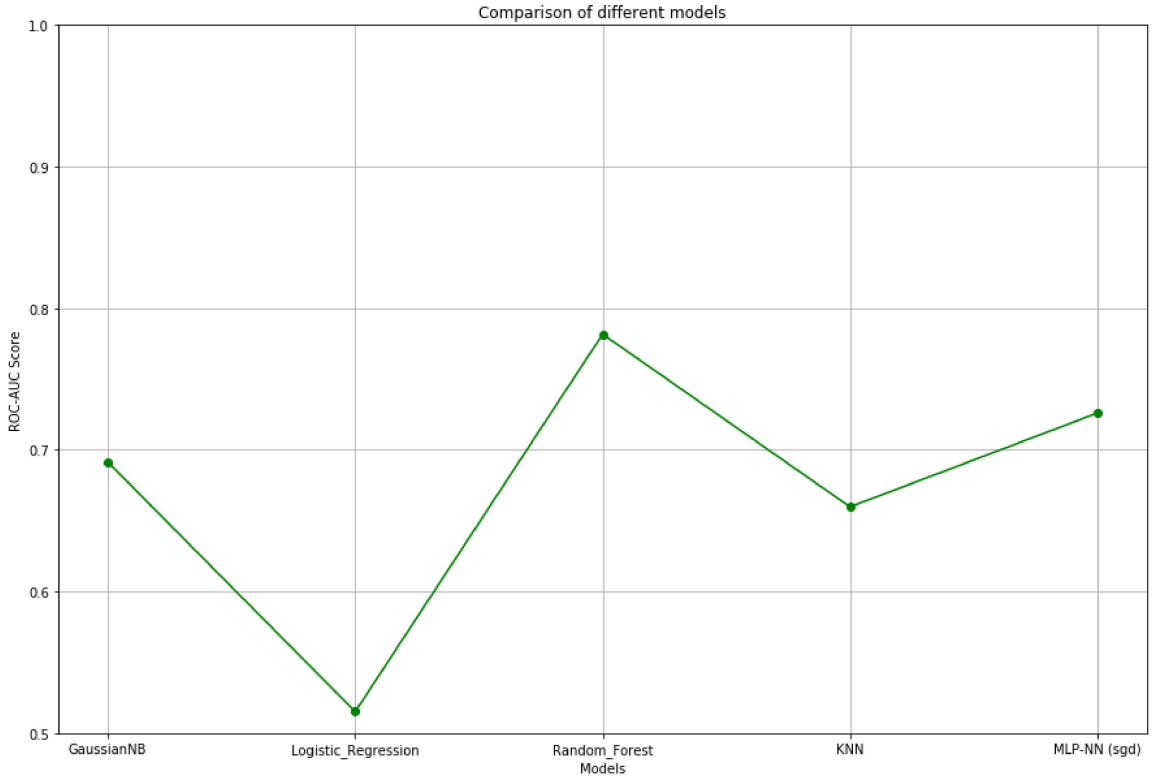
In [ ]:

```
#Model_Name.append('SVC')
#Model_Score.append(round(roc_auc_score(y_test, svc_pred), 4))
```

Model Comparison

In [372]:

```
plt.figure(figsize = (15, 10))
plt.plot(Model_Name,Model_Score, marker = 'o', color = 'green')
plt.title('Comparison of different models')
plt.xlabel('Models')
plt.ylabel('ROC-AUC Score')
plt.ylim(0.5, 1.0)
plt.grid()
plt.savefig('Model_comparison.jpeg')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
#df_clean
```

In [ ]:

```
# creates a function that will use TFIDF vectorizer to vectorize all text data
#from sklearn.feature_extraction.text import TfidfVectorizer

# initialize the vectorizer
#vectorizer = TfidfVectorizer()

#def vectorize_data(column):
    #df_clean[column] = list(vectorizer.fit_transform(df_clean[column]))
#text_columns = ['title', 'description', 'employment_type','required_experience','requi
red_education','industry','function', 'country_code', 'city']

#Loop through the text columns and vectorize each
#for column in text_columns:
    #vectorize_data(column)
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```