In [1]:

```
### Importing the requried libraries ###
```

In [2]:

```
import nltk
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [3]:

```
### Importing the data set ###
```

In [4]:

```
mnc = pd.read_csv("C:/Users/naikc/Downloads/Job titles and industries.csv")
```

Exploratory Data Analysis

In [5]:

```
### Finding the first 10 columns and row's of the data set ###
```

In [6]:

```
mnc.head(10)
```

Out[6]:

	Job Title	Industry
0	technical support and helpdesk supervisor - co...	IT
1	senior technical support engineer	IT
2	head of it services	IT
3	js front end engineer	IT
4	network and telephony controller	IT
5	privileged access management expert	IT
6	devops engineers x 3 - global brand	IT
7	devops engineers x 3 - global brand	IT
8	data modeller	IT
9	php web developer £45,000 based in london	IT

In [7]:

```
### Finding the last 10 columns & row's of the given data set ###
```

In [8]:

```
mnc.tail(10)
```

Out[8]:

	Job Title	Industry
8576	marketing & social media specialist	Marketing
8577	senior php developer	Marketing
8578	social media graphic designer	Marketing
8579	sponsorship sales executive	Marketing
8580	marketing specialist	Marketing
8581	data entry clerk	Marketing
8582	content creator	Marketing
8583	sales & marketing manager	Marketing
8584	marketing & digital marketing consultant	Marketing
8585	creative copywriter (arabic/english)	Marketing

In [9]:

```
### Find the total number of colmun's & row's in the data set ###
```

In [10]:

```
mnc.shape
```

Out[10]:

```
(8586, 2)
```

Out[10]: (8586, 2)

In [11]:

```
### Find the data types of the given data set ###
```

In [12]:

```
mnc.dtypes
```

Out[12]:

```
Job Title    object
Industry     object
dtype: object
```

In [13]:

```
mnc.describe()
```

Out[13]:

	Job Title	Industry
count	8586	8586
unique	3890	4
top	marketing executive	IT
freq	91	4746

In [14]:

```
### Finding the Null values or the missing values in data set ###
```

In [15]:

```
mnc.isnull().sum()
```

Out[15]:

```
Job Title    0
Industry     0
dtype: int64
```

In [16]:

```
### Find the total number of counts in each variable of the data set ###
```

In [17]:

```
mnc.Industry.value_counts()
```

Out[17]:

```
IT             4746
Marketing      2031
Education      1435
Accountancy     374
Name: Industry, dtype: int64
```

In [18]:

```
### Find the total number of counts in each variable of the data set ###
```

In [19]:

```
mnc["Job Title"].value_counts()
```

Out[19]:

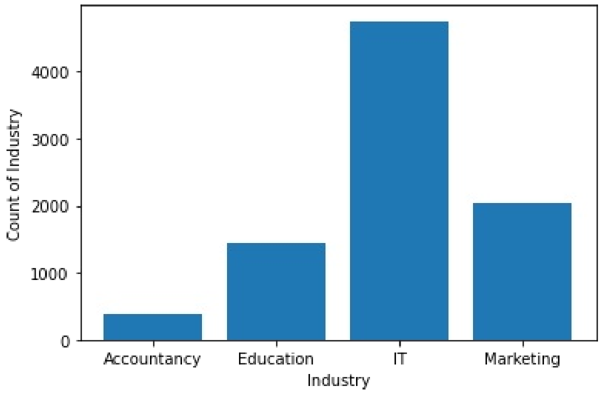
```
marketing executive                                  91
php developer                                        54
software developer                                   53
trainee network technician                           53
marketing manager                                    49
                                                     ..
sport graduate pe cover supervisors: preston - asap   1
seo account manager                                   1
paid media analyst                                    1
senior customer service specialist                    1
junior bi reporting analyst / support co-ordinator    1
Name: Job Title, Length: 3890, dtype: int64
```

In [20]:

```
### Visualisation of the data set between Industry & Job Title variables ###
```

In [21]:

```
Industry_count = mnc.groupby('Industry').count()
plt.bar(Industry_count.index.values, Industry_count['Job Title'])
plt.xlabel('Industry')
plt.ylabel('Count of Industry')
plt.show()
```



Creating some functions

In [22]:

```
def cleaner(text):
    text = text.lower()
    text = re.sub("german[^\s]+","",text)
    text = re.sub("bournemouth[^\s]+","",text)
    text = re.sub("international[^\s]+","",text)
    text = re.sub("flex[^\s]+","",text)
    text = re.sub("15[^\s]+","",text)
    text = re.sub("flexible[^\s]+","",text)
    text = re.sub("numerous[^\s]+","",text)
    text = re.sub("belfast[^\s]+","",text)
    text = re.sub("on[^\s]+","",text)
    text = re.sub("in[^\s]+","",text)
    text = re.sub("up[^\s]+","",text)
    text = re.sub("45[^\s]+","",text)
    text = re.sub("west[^\s]+","",text)
    text = re.sub("london[^\s]+","",text)
    text = re.sub("part[^\s]+","",text)
    text = re.sub("must[^\s]+","",text)
    text = re.sub("2[^\s]+","",text)
    text = re.sub("1/2[^\s]+","",text)
    text = re.sub("no[^\s]+","",text)
    text = re.sub("Â[^\s]+","",text)
    text = re.sub("12[^\s]+","",text)
    text = text.replace("1st","")
    text = re.sub("leading [^\s]+","",text)
    text = re.sub("1st[^\s]+","",text)
    text = re.sub("3rd[^\s]+","",text)
    text = re.sub("2nd[^\s]+","",text)
    text = re.sub("bristol[^\s]+","",text)
    text = re.sub("healthcare[^\s]+","",text)
    text = re.sub("good[^\s]+","",text)
    text = re.sub("pool[^\s]+","",text)
    text = re.sub("6 months[^\s]+","",text)
    text = re.sub("free[^\s]+","",text)
    text = re.sub("invest[^\s]+","",text)
    text = text.replace("o365","")
    text = text.replace("remote","")
    text = text.replace("-"," ")
    text = text.replace("/"," ")
    text = text.replace("("," ")
    text = text.replace(")"," ")
    text = text.replace("soa04086"," ")
    return text
```

In [23]:

```
### Removing the commas, semi colons, & slash's ###
```

In [24]:

```
def remove_stop_words(text):
    sw = stopwords.words("english")
    clean_words = []
    text = text.split()
    for word in text:
        if word not in sw:
            clean_words.append(word)
    return " ".join(clean_words)
```

In [25]:

```
### Stemming process or changing the words ###
```

In [26]:

```
def stemming(text):
    ps = PorterStemmer()
    text = text.split()
    stemmed_words = []
    for word in text :
        stemmed_words.append(ps.stem(word))
    return " ".join(stemmed_words)
```

In [27]:

```
### Running the changed words for the given data set ###
```

In [28]:

```
def run(text):
    text = cleaner(text)
    text = remove_stop_words(text)
    text = stemming(text)
    return text
```

In [29]:

```
### Checking with the Job Title variable ###
```

In [30]:

```
mnc['Job Title'] = mnc['Job Title'].apply(run)
```

In [31]:

```
### Checking with the first 10 columns of the given data set ###
```

In [32]:

```
mnc.head(10)
```

Out[32]:

	Job Title	Industry
0	technic helpdesk counti build ayr	IT
1	senior technic eng	IT
2	head servic	IT
3	js fr end eng	IT
4	network teleph c	IT
5	privileg access manag expert	IT
6	devop eng x 3 global brand	IT
7	devop eng x 3 global brand	IT
8	data model	IT
9	php web develop £ base l	IT

In [33]:

```
### Converting words to vector ###
```

In [34]:

```
tfidf = TfidfVectorizer()
x = tfidf.fit_transform(mnc["Job Title"]).toarray()
```

In [35]:

```
mnc['Industry'] = mnc['Industry'].replace("IT",0)
mnc['Industry'] = mnc['Industry'].replace("Marketing",1)
mnc['Industry'] = mnc['Industry'].replace("Education",2)
mnc['Industry'] = mnc['Industry'].replace("Accountancy",3)
```

In [36]:

```
y = mnc['Industry'].values
y
```

Out[36]:

```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

Splitting the data set

In [37]:

```
### Splitting up the data set into Train & Test data set respectivelly ###
```

In [38]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
```

Model Buliding

In [39]:

```
### Checking the Logistic Regression Model on data set ###
```

In [40]:

```
LR = LogisticRegression()
LR.fit(x_train,y_train)
```

Out[40]:

```
LogisticRegression()
```

In [41]:

```
### Checking the prediction on x_train ###
```

In [42]:

```
y_pred = LR.predict(x_test)
```

In [43]:

```
### Checking for the RMSE value for x_train
```

In [44]:

```
LR.score(x_test,y_test)
```

Out[44]:

```
0.9105728924080112
```

In [45]:

```
### Checking for different models ###
### Importing their respective libraries ###
```

In [46]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [47]:

```
### Splitting the data set & checking their RMSE values ###
```

In [48]:

```
clfs = [GaussianNB(),SVC(kernel="linear"),SVC(kernel="rbf"),DecisionTreeClassifier(),RandomForestClassifier(n_est
for clf in clfs:
    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    print("===================",clf)

    ### print(clf.score(x_test,y_test)*100) ###

    print(clf.score(x_test,y_test))
```

```
=================== GaussianNB()
0.6390312063344201
=================== SVC(kernel='linear')
0.9142990218910108
=================== SVC()
0.9226828132277597
=================== DecisionTreeClassifier()
0.9073125291103866
=================== RandomForestClassifier()
0.9133674895202608
```

Chosen Model

In [49]:

```
### I have chosen Random Forest as my Final Model as it was giving a good RMSE value ###
```

In [50]:

```
model = RandomForestClassifier(n_estimators=100)
model.fit(x_train,y_train)
```

Out[50]:

```
RandomForestClassifier()
```

In [51]:

```
### Checking for the prediction ###
```

In [52]:

```
y_pred = LR.predict(x_test)
```

In [53]:

```
### Checking for the RMSE value ###
```

In [54]:

```
model.score(x_test,y_test)
```

Out[54]:

```
0.9152305542617606
```

Model Serialization

In [55]:

```
### Imoprting the requried library ###
```

In [56]:

```
import pickle
```

In [57]:

```
### Saving the Model ###
```

In [58]:

```
pickle.dump(model, open('FinalModel', 'wb'))
```

In [59]:

```
### Loading model to compare the results ###
```

In [60]:

```
model = pickle.load(open('FinalModel', 'rb'))
```

Testing

In [61]:

```
test = "data modeller"
test = run(test)
test = tfidf.transform([test]).toarray()
```

In [62]:

```
model.predict(test)
```

Out[62]:

```
array([0], dtype=int64)
```

In [63]:

```
test2 = "analytics manager"
test2 = run(test2)
test2 = tfidf.transform([test2]).toarray()
```

In [64]:

```
model.predict(test2)
```

Out[64]:

```
array([1], dtype=int64)
```

In [65]:

```
test3 = "careers advisor"
test3 = run(test3)
test3 = tfidf.transform([test3]).toarray()
```

In [66]:

```
model.predict(test3)
```

Out[66]:

```
array([2], dtype=int64)
```

In [67]:

```
test4 ="credit controller"
test4 = run(test4)
test4 = tfidf.transform([test4]).toarray()
```

In [68]:

```
model.predict(test4)
```

Out[68]:
```
array([3], dtype=int64)
```

In [69]:
```
test5 = "finance assistant"
test5 = run(test5)
test5 = tfidf.transform([test5]).toarray()
```

In [70]:
```
model.predict(test5)
```

Out[70]:
```
array([3], dtype=int64)
```

Model Evaluation

In [71]:
```
from sklearn.metrics import classification_report
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.91      0.94      1261
           1       0.88      0.90      0.89       507
           2       0.83      0.95      0.89       310
           3       0.64      0.81      0.71        69

    accuracy                           0.91      2147
   macro avg       0.83      0.89      0.86      2147
weighted avg       0.92      0.91      0.91      2147
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js