Job title Classification by industry

Importing Libraries

In [1]:

```
## for data
import json
import pandas as pd
import numpy as np
import os

## for plotting
import matplotlib.pyplot as plt
import seaborn as sns

## for processing
from sklearn.utils.class_weight import compute_sample_weight
import nltk.corpus
from gensim.utils import simple_preprocess
from sklearn.feature_extraction.text import TfidfVectorizer
lst_stopwords = nltk.corpus.stopwords.words("english")
lst_stopwords.remove('it')
from imblearn.over_sampling import SMOTE

## For Model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier


## For Evaluation
from sklearn.metrics import precision_score,recall_score,accuracy_score,f1_score,roc_au
```

Functions

In [2]:

```
def title_preprocess(title,lst_stopwords):
    ## clean (convert to lowercase and remove punctuations and characters and then stri
    lst_text = simple_preprocess(title)

    ## remove Stopwords
    if lst_stopwords is not None:
        lst_text = [word for word in lst_text if word not in
```

```
                    lst_stopwords]

    ## back to string from list
    text = " ".join(lst_text)
    return text
```

Get the Data¶

In [4]:
```
DATASET_PATH = './Dataset/'
Data = pd.read_csv(os.path.join(DATASET_PATH, 'Job titles and industries.csv'))
print(Data.shape)
Data.head()
```

(8586, 2)

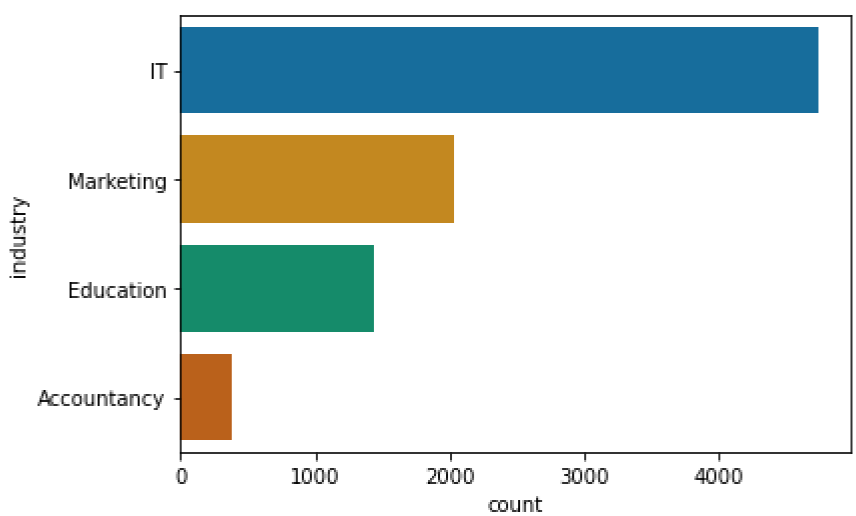
Out[4]:

	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

Explore the Data

a) The Distribution of the data

In [5]:
```
sns.countplot(y = "industry",palette = "colorblind",data = Data,);
```



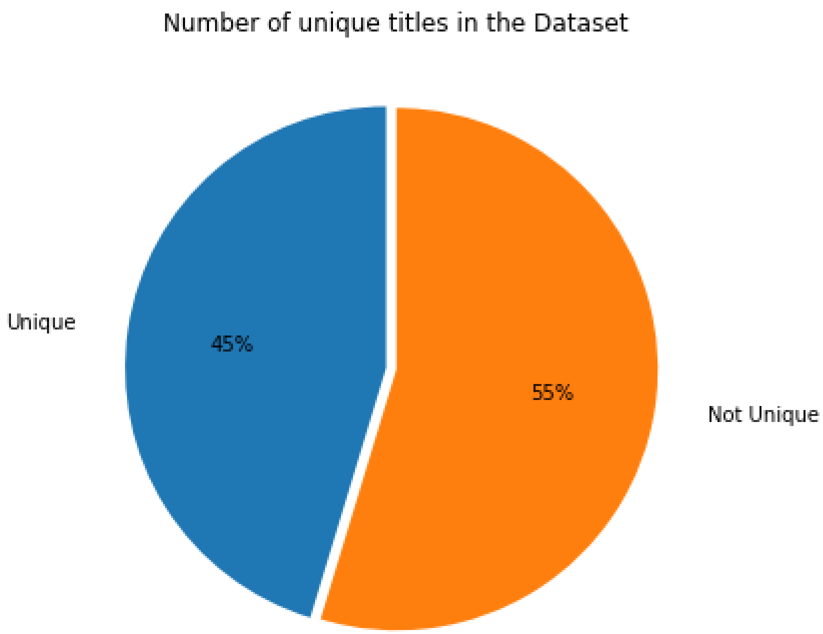
As you can see, datasets that are used for classification have imbalanced classes issue. If we train a multi classification model without fixing this problem, the model will be completely biased.

b) Unique titles in the dataset

In [6]:
```
###Bar plot for the number of unique titles in the dataset
all_unique=Data['job title'].nunique()
```

```
##Calculating Percentages
uniq_per=(all_unique/Data.shape[0])*100
not_per=((Data.shape[0]-all_unique)/Data.shape[0])*100
print("Percentage of unique Questions: ",np.round(uniq_per,2))

##Pie Chart
fig = plt.subplots(figsize =(10, 6));
plt.pie([uniq_per,not_per],labels=["Unique", "Not Unique"],startangle=90,autopct='%1.0f
plt.title("Number of unique titles in the Dataset");
```

```
Percentage of unique Questions:  45.31
```



Data & Text cleaning

In [7]:

```
Data["title_cleaned"] = Data["job title"].apply(lambda x:title_preprocess(x, lst_stopwo

vectorizer = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1'
vectorizer.fit(Data["title_cleaned"])
Tfidf = vectorizer.transform(Data["title_cleaned"])
Tfidf_Data=pd.DataFrame(Tfidf.toarray(),columns=np.array(vectorizer.get_feature_names()
```

Handling imbalanced data useing Oversampling technique on data

It's generating synthetic data that tries to randomly generate a sample of the attributes from observations in the minority class.

In [8]:

```
X = Tfidf_Data
y = Data['industry']

## handling imbalanced data
SMOTE = SMOTE()
```

```
X_OS, y_OS = SMOTE.fit_resample(Tfidf_Data, y)

X_train, X_test, y_train, y_test = train_test_split(X_OS , y_OS, test_size=0.2, random_
```

In [9]:

```
y_train.value_counts()/len(y_train)
```

Out[9]:

```
IT             0.251136
Accountancy    0.249951
Education      0.249687
Marketing      0.249226
Name: industry, dtype: float64
```

In [10]:

```
y_test.value_counts()/len(y_test)
```

Out[10]:

```
Marketing      0.253095
Education      0.251251
Accountancy    0.250198
IT             0.245457
Name: industry, dtype: float64
```

Model

We will try different Classifiers to see best model for our problem

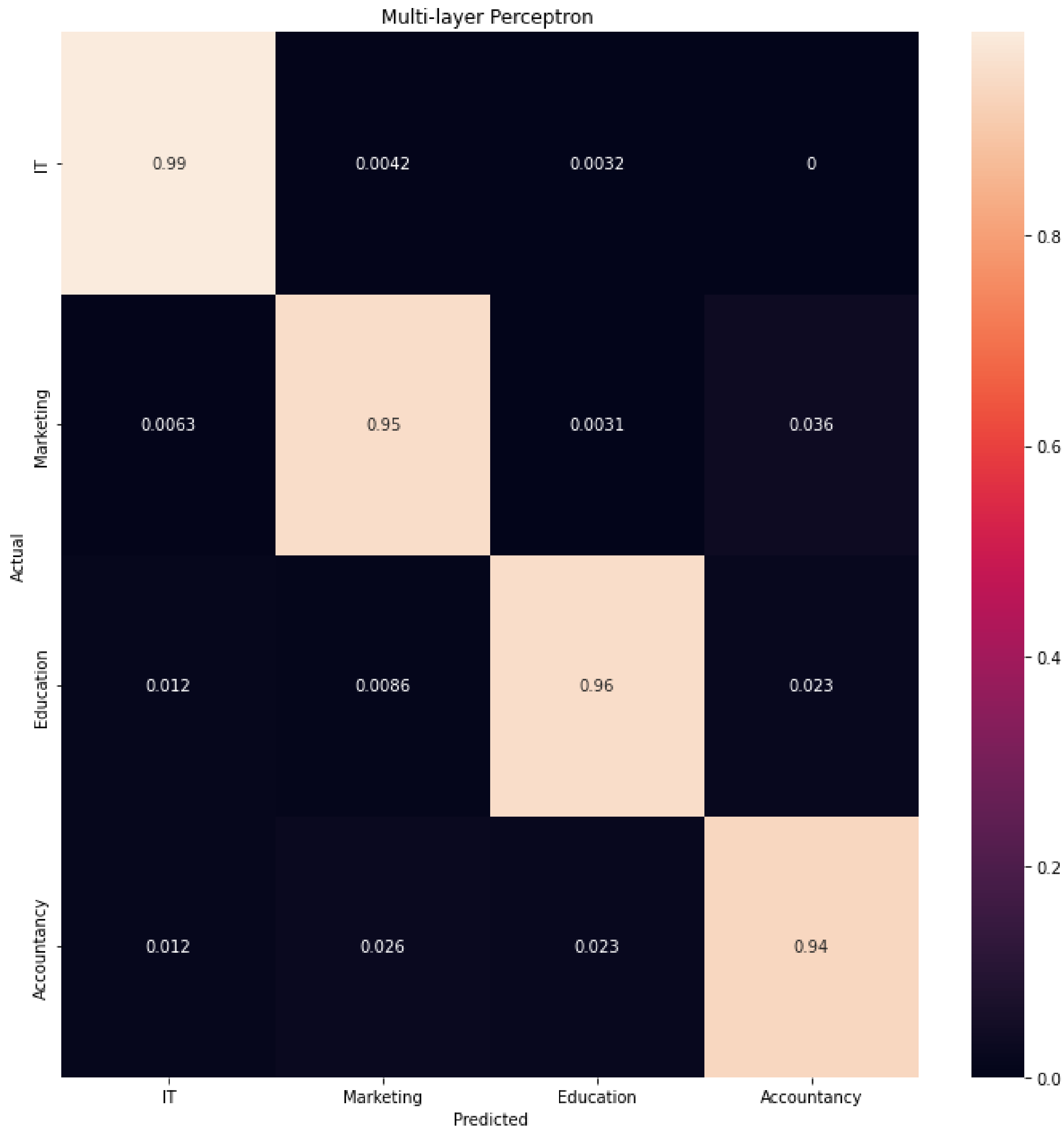
1) Multi-layer Perceptron

In [11]:

```
MLP = MLPClassifier(
    hidden_layer_sizes=(50,),
    batch_size=50,
    max_iter=500,
    shuffle=False,
    random_state=0,
    warm_start=True,
)
MLP.fit(X_train, y_train) # fit the model
y_MLP = MLP.predict(X_test)

#evaluate the model
print("classification report:\n\n",classification_report(y_test, y_MLP))

conf_mat = confusion_matrix(y_test, y_MLP,normalize='true')
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(conf_mat, annot=True,xticklabels=list(Data['industry'].unique()),yticklabel
plt.title("Multi-layer Perceptron ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
classification report:

               precision    recall  f1-score   support

 Accountancy       0.97      0.99      0.98       950
   Education       0.96      0.95      0.96       954
          IT       0.97      0.96      0.96       932
```

```
   Marketing       0.94      0.94      0.94       961

    accuracy                           0.96      3797
   macro avg       0.96      0.96      0.96      3797
weighted avg       0.96      0.96      0.96      3797
```

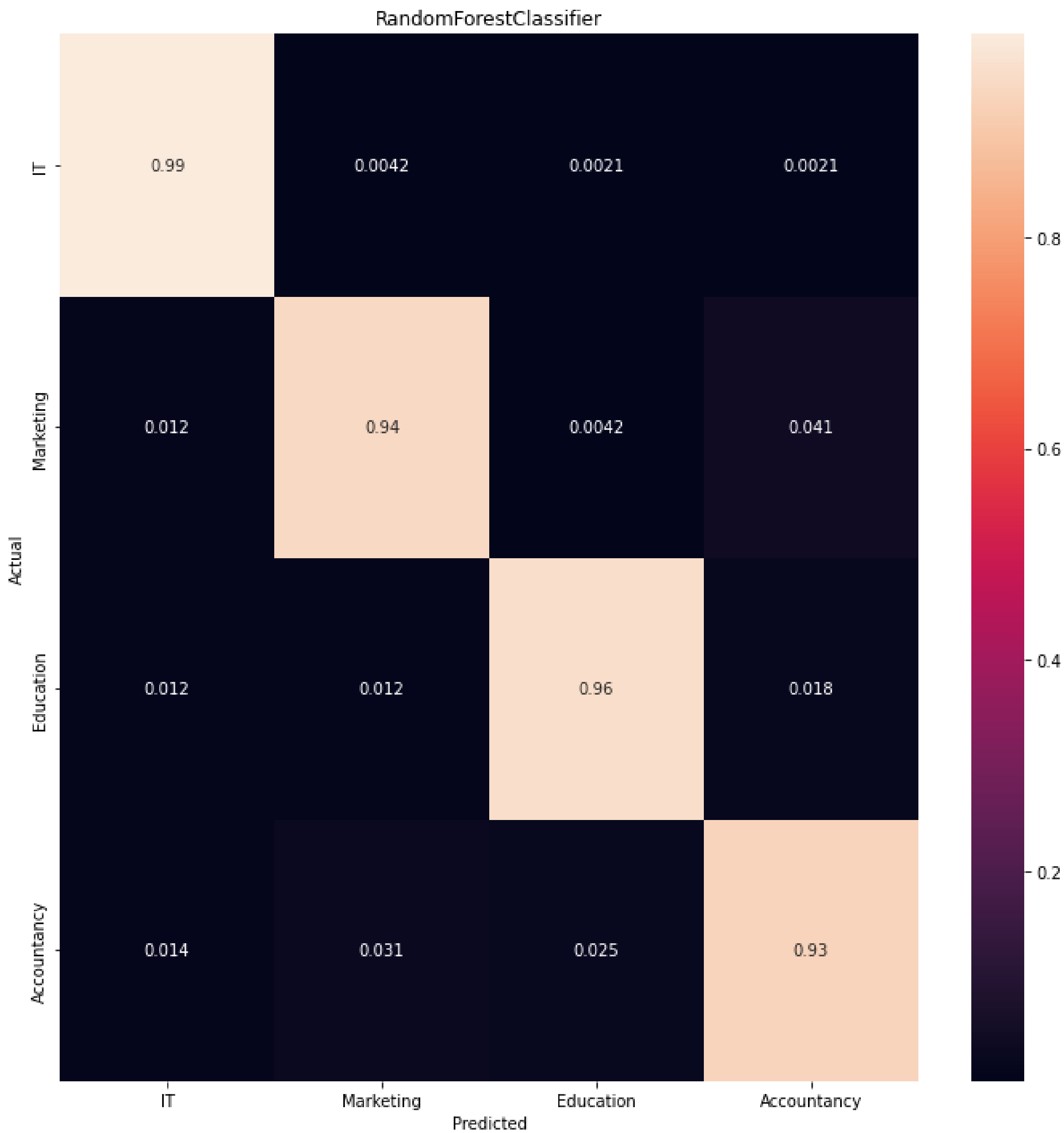


2) Decision Tree

In [12]:
```
rfc=RandomForestClassifier(n_estimators=30,random_state=42)
rfc.fit(X_train, y_train)  # fit the model
y_rfc=rfc.predict(X_test)

#evaluate the model
print("classification report:\n\n",classification_report(y_test, y_rfc))
conf_mat = confusion_matrix(y_test, y_rfc,normalize='true')
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(conf_mat, annot=True,xticklabels=list(Data['industry'].unique()),yticklabel
```

```
plt.title("RandomForestClassifier")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
classification report:

              precision    recall  f1-score   support

 Accountancy       0.96      0.99      0.98       950
   Education       0.95      0.94      0.95       954
          IT       0.97      0.96      0.96       932
   Marketing       0.94      0.93      0.93       961

    accuracy                           0.96      3797
   macro avg       0.96      0.96      0.96      3797
weighted avg       0.96      0.96      0.96      3797
```

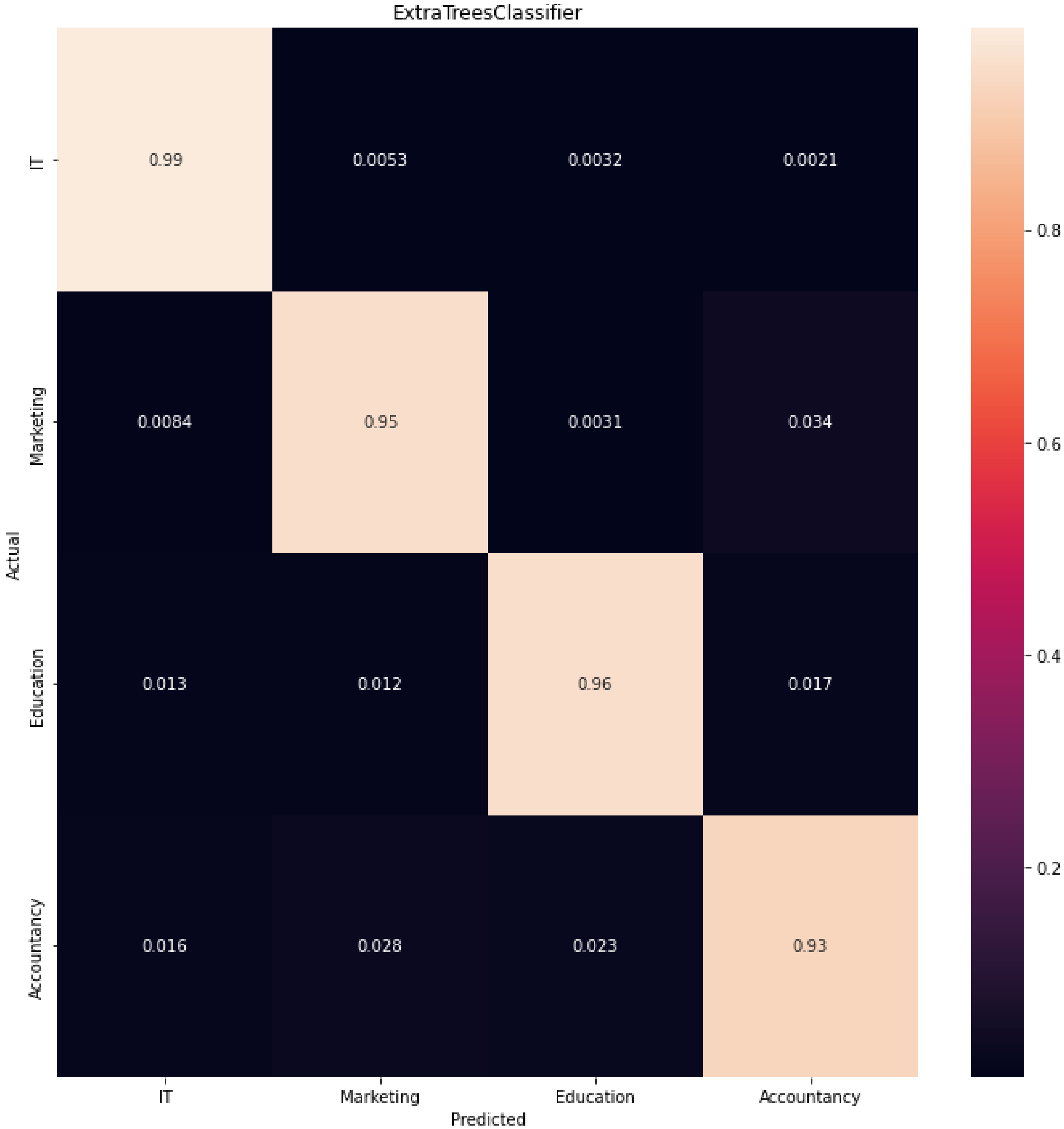


3) ExtraTreesClassifier Model

In [13]:

```
xtc = ExtraTreesClassifier(n_estimators=30, random_state=42)
xtc.fit(X_train, y_train)
y_xtc=xtc.predict(X_test)
print("classification report:\n\n",classification_report(y_test, y_xtc))

conf_mat = confusion_matrix(y_test, y_xtc,normalize='true')
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(conf_mat, annot=True,xticklabels=list(Data['industry'].unique()),yticklabel
plt.title("ExtraTreesClassifier")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
classification report:

               precision    recall  f1-score   support

 Accountancy       0.96      0.99      0.98       950
   Education       0.95      0.95      0.95       954
          IT       0.97      0.96      0.96       932
   Marketing       0.95      0.93      0.94       961

    accuracy                           0.96      3797
   macro avg       0.96      0.96      0.96      3797
weighted avg       0.96      0.96      0.96      3797
```



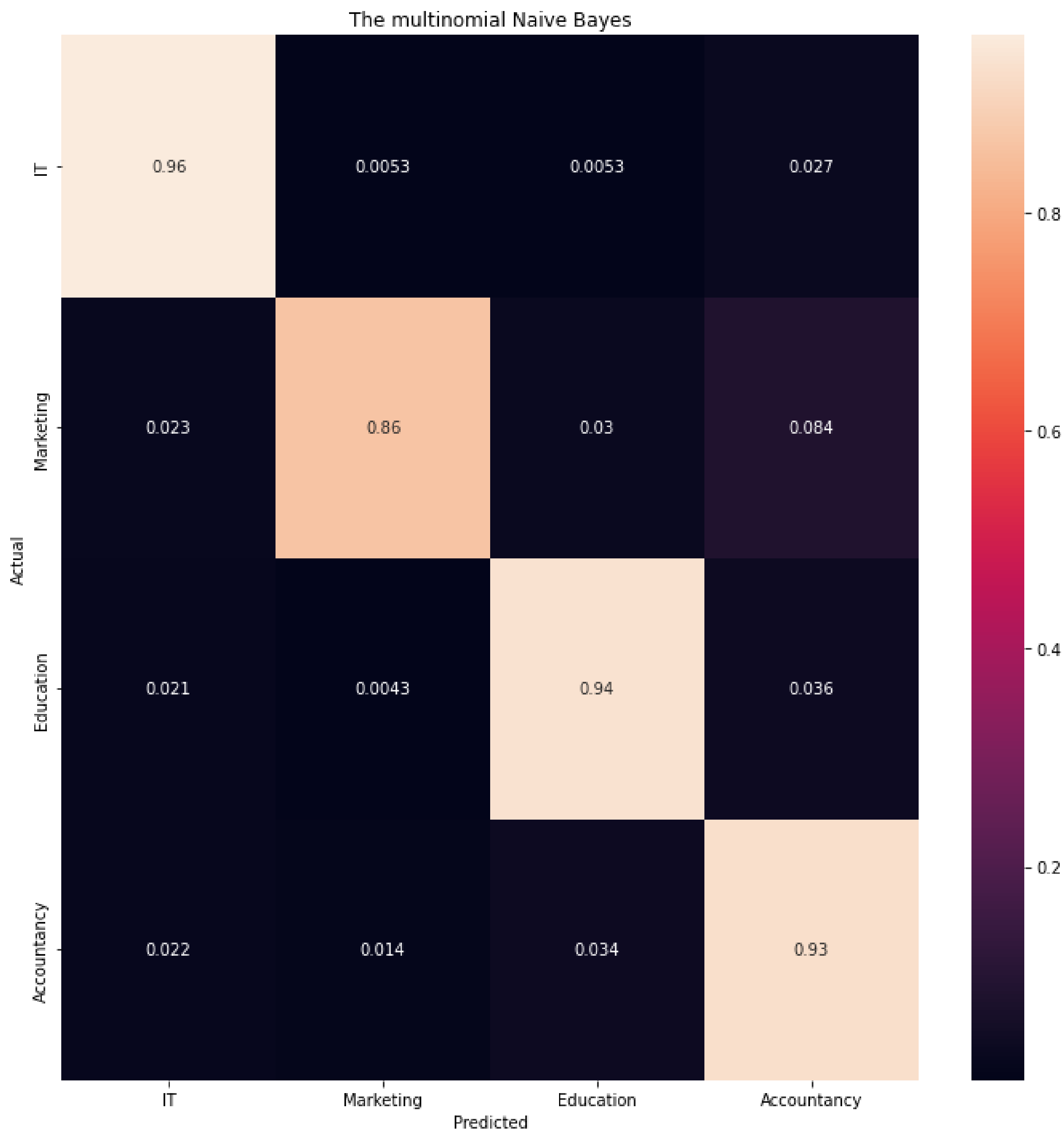
4) The multinomial Naive Bayes

In [15]:

```
MulNB=MultinomialNB()
MulNB.fit(X_train,y_train)
y_MulNB=MulNB.predict(X_test)
print("classification report:\n\n",classification_report(y_test, y_MulNB))
conf_mat = confusion_matrix(y_test, y_MulNB,normalize='true')
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(conf_mat, annot=True,xticklabels=list(Data['industry'].unique()),yticklabel
plt.title("The multinomial Naive Bayes")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
classification report:

               precision    recall  f1-score   support
```

```
 Accountancy       0.94      0.96      0.95       950
   Education       0.97      0.86      0.91       954
          IT       0.93      0.94      0.93       932
   Marketing       0.86      0.93      0.90       961

    accuracy                           0.92      3797
   macro avg       0.93      0.92      0.92      3797
weighted avg       0.93      0.92      0.92      3797
```



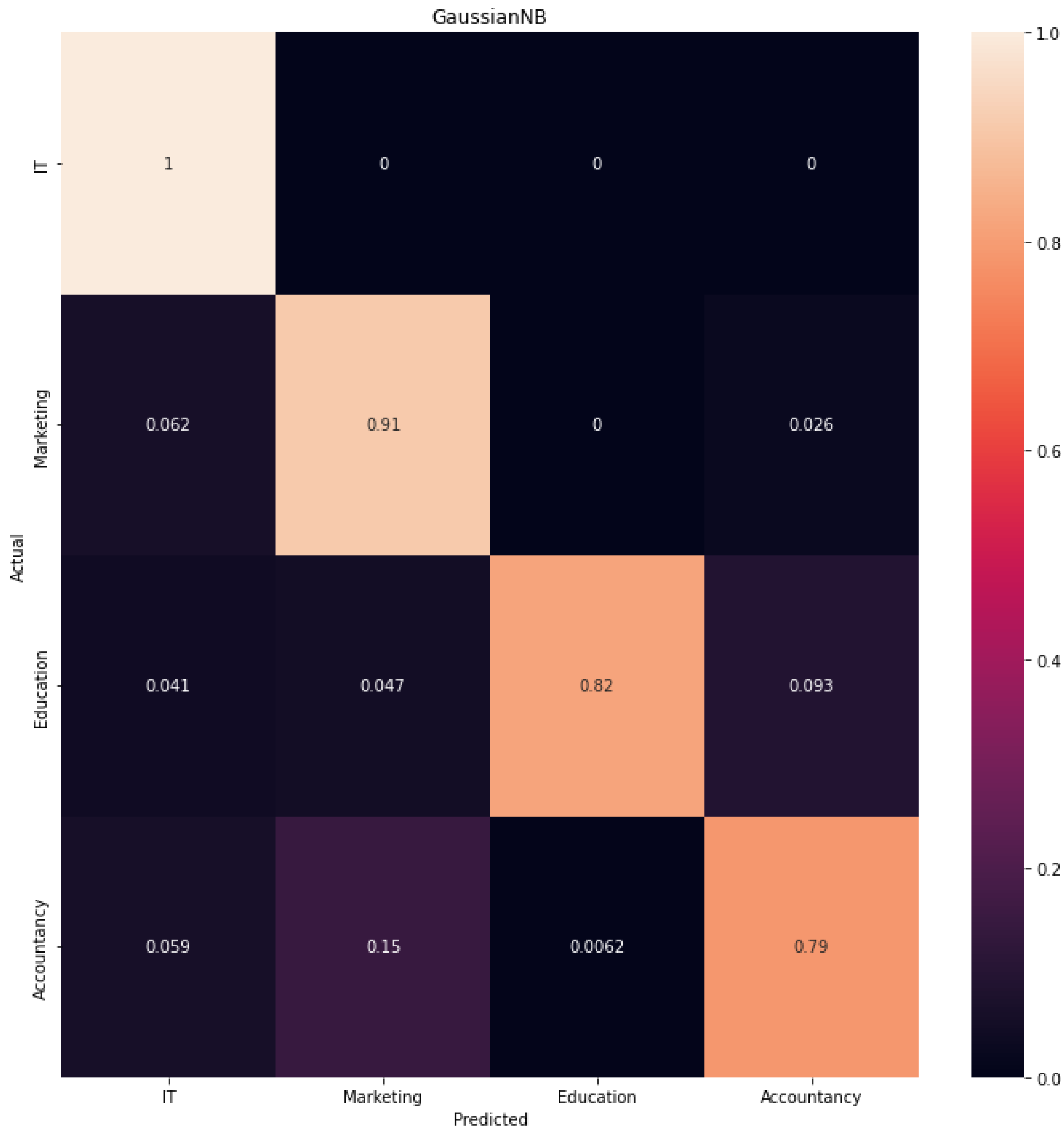
4) The Gaussian Naive Bayes

In [17]:

```
gnb=GaussianNB()
gnb.fit(X_train,y_train)
y_gnb=gnb.predict(X_test)
print("classification report:\n\n",classification_report(y_test, y_gnb))
conf_mat = confusion_matrix(y_test, y_gnb,normalize='true')
```

```
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(conf_mat, annot=True,xticklabels=list(Data['industry'].unique()),yticklabel
plt.title("GaussianNB")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
classification report:

               precision    recall  f1-score   support

 Accountancy       0.86      1.00      0.93       950
   Education       0.82      0.91      0.87       954
          IT       0.99      0.82      0.90       932
   Marketing       0.87      0.79      0.83       961

    accuracy                           0.88      3797
   macro avg       0.89      0.88      0.88      3797
weighted avg       0.89      0.88      0.88      3797
```



Final Model VotingClassifier

In [19]:
```
classifiers = [

    ('GaussianNB', gnb),
    ('MultinomialNB', MulNB),

    ('ExtraTreesClassifier', xtc),
    ('RandomForest', rfc),
    #('KNeighborsClassifier', KN),
    #('DecisionTreeClassifier', dtc),



]
```

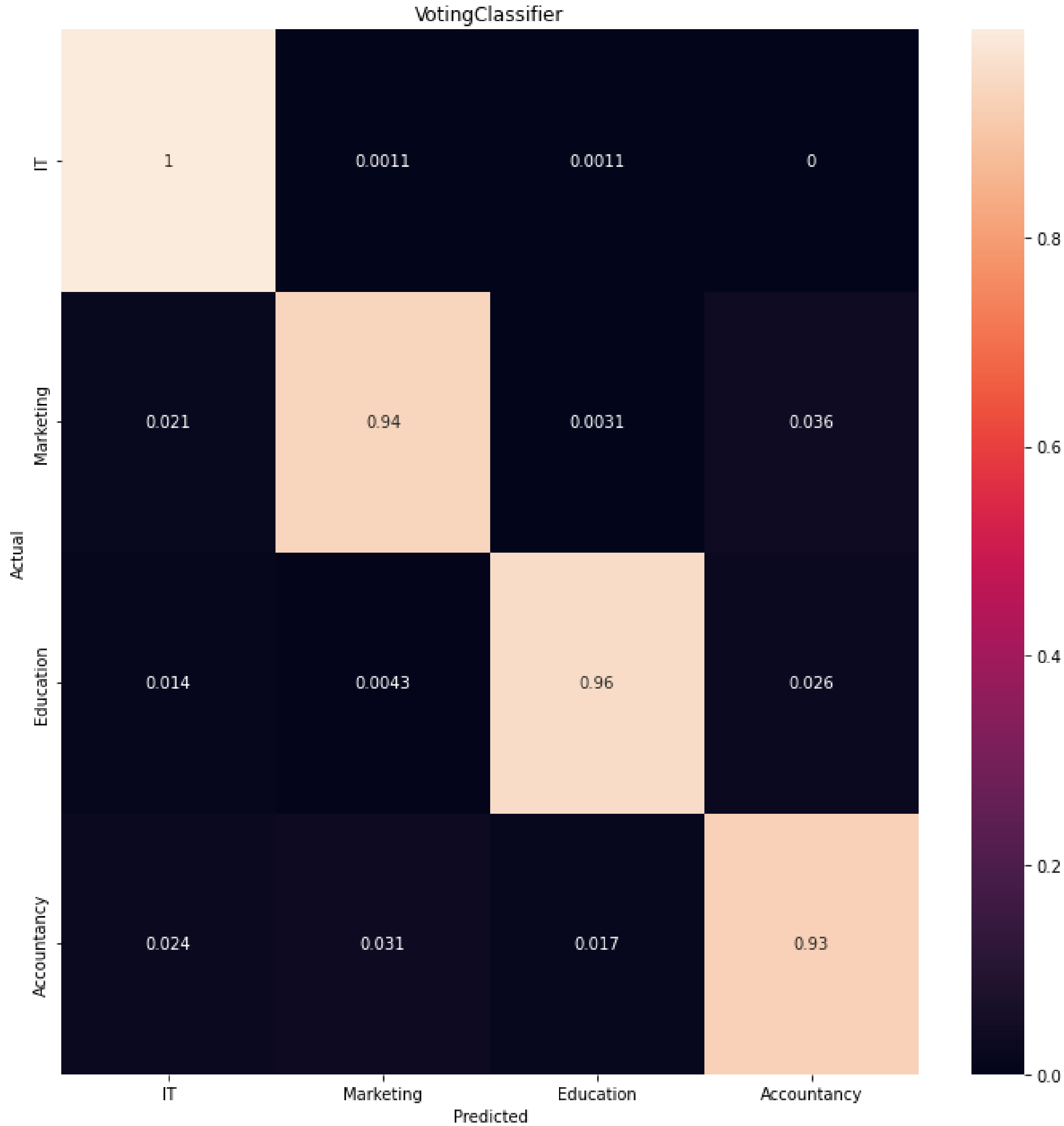
```
pipe = Pipeline([('classifier', VotingClassifier(estimators=classifiers,voting='soft'))

pipe.fit(X_train, y_train)
print('Training set score: ' + str(pipe.score(X, y)))
```

```
Training set score: 0.9614488702539017
```

In [20]:

```
y_F = pipe.predict(X_test)
conf_mat = confusion_matrix(y_test, y_F,normalize='true')
print("classification report:\n\n",classification_report(y_test, y_F))

fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(conf_mat, annot=True,xticklabels=list(Data['industry'].unique()),yticklabel
plt.title("VotingClassifier")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
classification report:

               precision    recall  f1-score   support

 Accountancy       0.94      1.00      0.97       950
   Education       0.96      0.94      0.95       954
          IT       0.98      0.96      0.97       932
   Marketing       0.94      0.93      0.93       961

    accuracy                           0.96      3797
   macro avg       0.96      0.96      0.96      3797
weighted avg       0.96      0.96      0.96      3797
```



In [21]:

```
import pickle
pickle.dump(pipe, open('model.pkl','wb'))
```

References

https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794
https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794
https://www.youtube.com/playlist?list=PLQpVVU8sLpqx0t7aFOfXJzJMrtxX5yvKU

In [ ]: