KNN-after-data-balanced-and-numeralization

December 10, 2022

[1]:
```python
# library
import math
import random
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.utils import resample
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

[2]:
```python
# load cleaned data
data = pd.read_pickle("cleaned.pkl")
```

[3]:
```python
data
```

[3]:
```
                                         company_profile  \
0      food52 weve created groundbreaking awardwinnin…
1      90 seconds worlds cloud video production servi…
2      valor services provides workforce solutions me…
3      passion improving quality life geography heart…
4      spot source solutions llc global human capital…
…                                                    …
17875  vend looking awesome new talent come join us y…
17876  web linc ecommerce platform services provider …
17877  provide full time permanent positions many med…
17878                                                 na
17879  Vend is looking for some awesome new talent to…

                                             description  \
0      food52 fastgrowing james beard awardwinning on…
1      organised focused vibrant awesomedo passion cu…
2      client located houston actively seeking experi…
3      company esri environmental systems research in…
4      job title itemization review manager location …
…                                                    …
17875  case first time youve visited website vend awa…
```

```
17876  payroll accountant focus primarily payroll fun…
17877  experienced project cost control staff enginee…
17878  nemsia studios looking experienced visualgraph…
17879  wevend award winning web based point sale soft…

                                            requirements  \
0      experience content management systems major pl…
1      expect key responsibility communicate client 9…
2      implement precommissioning commissioning proce…
3      education bachelors masters gis business admin…
4      qualifications rn license state texas diploma …
…                                                    …
17875  ace role eat comprehensive statements work bre…
17876  ba bs accounting desire fun love genuine passi…
17877  least 12 years professional experienceability …
17878  1 must fluent latest versions corel amp adobe …
17879  We want to hear from you if:You have an in-dep…

                                                benefits  fraudulent
0      experience content management systems major pl…           0
1      expect key responsibility communicate client 9…           0
2      implement precommissioning commissioning proce…           0
3      education bachelors masters gis business admin…           0
4      qualifications rn license state texas diploma …           0
…                                                    …         …
17875  ace role eat comprehensive statements work bre…           0
17876  ba bs accounting desire fun love genuine passi…           0
17877  least 12 years professional experienceability …           0
17878  1 must fluent latest versions corel amp adobe …           0
17879                                                 NA           0

[17879 rows x 5 columns]
```

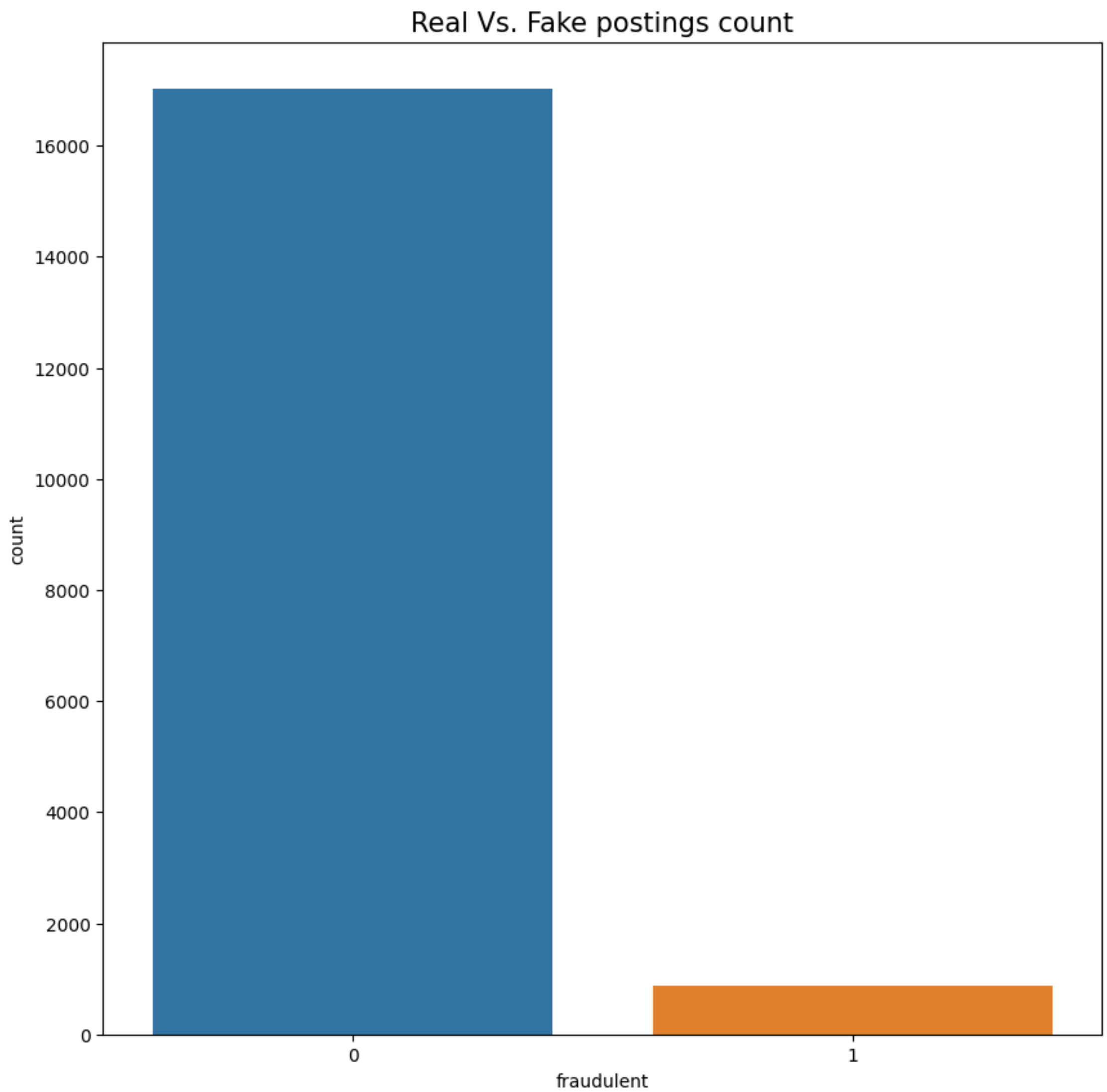
[4]:
```python
# We connect all text together as one feature.
data["full_text"] = data["company_profile"] + " " + data["description"] + " " +
    data["requirements"] + " " + data["benefits"]
```

[5]:
```python
# Check if there has any Null value
null_all = data.isnull().sum()
print(null_all)
```

```
company_profile    0
description        0
requirements       0
benefits           0
fraudulent         0
full_text          0
dtype: int64
```

1 Balance Data Set

[6]:
```
# Check if imbalance data by using bar figure.
plt.figure(figsize = (10,10))
sns.countplot(x="fraudulent", data=data)
plt.title("Real Vs. Fake postings count", fontsize = 15)
plt.show()
```



[7]:
```
train, test = train_test_split(data, test_size=0.2, random_state = 1)
```

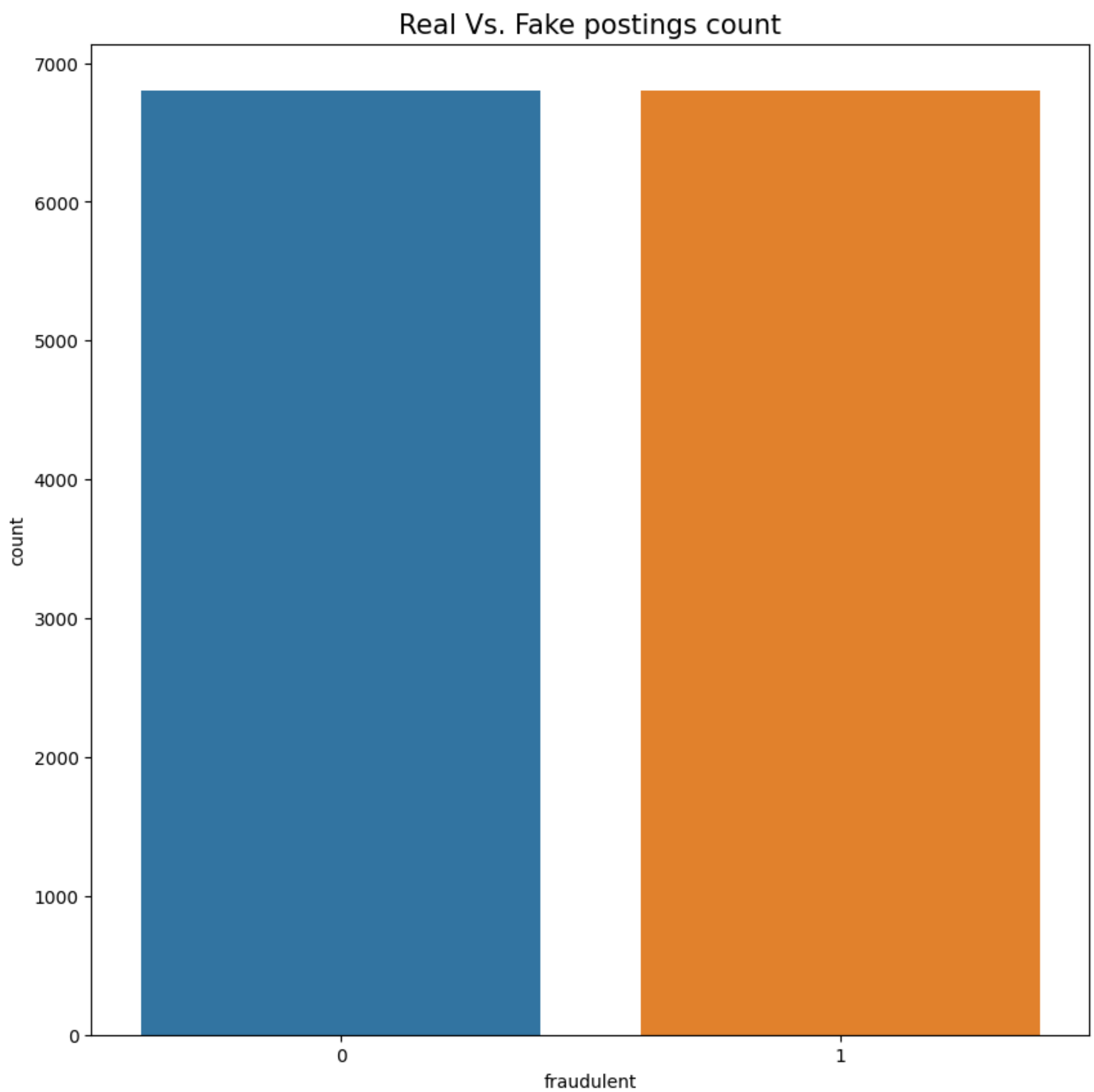
[8]:
```
# Check imbalance data distribution.
print("Number of cases: " , len(train))
print("Number of fraudulent cases: ", len(train[train["fraudulent"] == 1]))
```

```
print("Number of non fraudulent cases: ", len(train[train["fraudulent"] == 0]))
```

```
Number of cases:  14303
Number of fraudulent cases:  708
Number of non fraudulent cases:  13595
```

[9]:
```
# Random seed.
random.seed(1)

# Since the fraudulent cases is extremely less than non-fradulent cases, we assign non-fraudulent as majority.
df_majority = train[train["fraudulent"]== 0]
df_minority = train[train["fraudulent"]== 1]

# Upsample the dataset by simply copying records from minority classes by using resample().
# The value for the n_samples parameter is set to a half of the number of majority class to avoid overfitting.
negative_upsample = resample(df_minority, replace = True,
                        n_samples = math.ceil(df_majority.shape[0]/2),
                        random_state = 101)

# Aslo, we need to undersample majority classes
negative_undersample = resample(df_majority, replace = True,
                        n_samples = math.ceil(df_majority.shape[0]/2),
                        random_state = 101)

# Concat two dataframes (majority class and upsampled minority class).
df_upsampled = pd.concat([negative_undersample, negative_upsample])
df_upsampled = df_upsampled.sample(frac = 1, random_state = 101)
```

[10]:
```
# Show data distribution after resample
plt.figure(figsize = (10,10))
sns.countplot(x="fraudulent", data=df_upsampled)
plt.title("Real Vs. Fake postings count", fontsize = 15)
plt.show()
```



[11]:
```
# Check data after resample.
print("Number of cases: " , len(df_upsampled))
print("Number of fraudulent cases: ",␣
↪len(df_upsampled[df_upsampled["fraudulent"] == 1]))
print("Number of non fraudulent cases: ",␣
↪len(df_upsampled[df_upsampled["fraudulent"] == 0]))
```

```
Number of cases:  13596
Number of fraudulent cases:  6798
Number of non fraudulent cases:  6798
```

[12]:
```
# Set train data as "full_text" feature and set target value.
train_x = df_upsampled['full_text']
```

```
test_x = test['full_text']

train_y = df_upsampled['fraudulent']
test_y = test['fraudulent']
```

[13]:
```
# In order to perform machine learning on text documents, we first need to turn␣
↪the text content into numerical feature vectors.
# Our model cannot simply read the text data so we convert it into numerical␣
↪format.
# In order to convert the data into numerical format we create vectors from␣
↪text.
vectorizer = TfidfVectorizer()

train_data = vectorizer.fit_transform(train_x)
test_data = vectorizer.transform(test_x)
```

1.0.1 KNN implementation

For applying KNN, first import the KNeighborsClassifier module and create KNN classifier object by giving a number of neighbors in KNeighborsClassifier() function.

[14]:
```
# import needed libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Implement KNN on unprocessed dataset First, check the performance of KNN by using the oringinal uncleaned and unbalanced data, pick the 'description' as the feature.

[15]:
```
#define the variables
x = data['description']
y = data['fraudulent']

#split it into training and test sets
x_train_unprocessed, x_test_unprocessed, y_train_unprocessed,␣
↪y_test_unprocessed = train_test_split(x, y, test_size=0.2, random_state = 1)

#vectorization
vectorizer = TfidfVectorizer()
vectorizer.fit(x_train_unprocessed)

x_trainvec = vectorizer.transform(x_train_unprocessed)
x_testvec = vectorizer.transform(x_test_unprocessed)

knn = KNeighborsClassifier()
```

```
knn.fit(x_trainvec, y_train_unprocessed)
y_pred_unprocessed = knn.predict(x_testvec).flatten()
accuracy_score(y_pred_unprocessed, y_test_unprocessed)
```

[15]: 0.9658836689038032

Implement KNN on processed dataset In our implementation, we use K=1, K=3, and K=5 and compare their accuracy, confusion matrix, and finally check the difference between their classification reports.

[16]:
```
knn1 = KNeighborsClassifier(n_neighbors=1)
knn1.fit(train_data, train_y)

knn3 = KNeighborsClassifier(n_neighbors=3)
knn3.fit(train_data, train_y)

knn5 = KNeighborsClassifier(n_neighbors=5)
knn5.fit(train_data, train_y)
```

[16]: KNeighborsClassifier()

[17]:
```
y_test = test_y.values.flatten()

y_pred1 = knn1.predict(test_data).flatten()
y_pred3 = knn3.predict(test_data).flatten()
y_pred5 = knn5.predict(test_data).flatten()
```

[18]:
```
# report
knn_accuracy1 = accuracy_score(y_pred1, y_test)
knn_confusionMatrix1 = confusion_matrix(y_test, y_pred1)
knn_classification1 = classification_report(y_test, y_pred1)

knn_accuracy3 = accuracy_score(y_pred3, y_test)
knn_confusionMatrix3 = confusion_matrix(y_test, y_pred3)
knn_classification3 = classification_report(y_test, y_pred3)

knn_accuracy5 = accuracy_score(y_pred5, y_test)
knn_confusionMatrix5 = confusion_matrix(y_test, y_pred5)
knn_classification5 = classification_report(y_test, y_pred5)

# print report
print("- Accuracy score of KNN")
print(f"K=1: {knn_accuracy1}")
print(f"K=3: {knn_accuracy3}")
print(f"K=5: {knn_accuracy5}\n\n")

print("- Confusion matrix of KNN")
```

```
print(f"K=1:\n {knn_confusionMatrix1}\n")
print(f"K=3:\n {knn_confusionMatrix3}\n")
print(f"K=5:\n {knn_confusionMatrix5}\n\n")


print("- Classification report of KNN")
print(f"\nK=1:\n {knn_classification1}\n")
print(f"\nK=3:\n {knn_classification3}\n")
print(f"\nK=5:\n {knn_classification5}")
```

```
- Accuracy score of KNN
K=1: 0.9700782997762863
K=3: 0.9538590604026845
K=5: 0.9384787472035794


- Confusion matrix of KNN
K=1:
 [[3334   85]
 [  22  135]]

K=3:
 [[3274  145]
 [  20  137]]

K=5:
 [[3216  203]
 [  17  140]]


- Classification report of KNN

K=1:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      3419
           1       0.61      0.86      0.72       157

    accuracy                           0.97      3576
   macro avg       0.80      0.92      0.85      3576
weighted avg       0.98      0.97      0.97      3576



K=3:
              precision    recall  f1-score   support

           0       0.99      0.96      0.98      3419
```

```
           1       0.49      0.87      0.62       157

    accuracy                           0.95      3576
   macro avg       0.74      0.92      0.80      3576
weighted avg       0.97      0.95      0.96      3576



K=5:
              precision    recall  f1-score   support

           0       0.99      0.94      0.97      3419
           1       0.41      0.89      0.56       157

    accuracy                           0.94      3576
   macro avg       0.70      0.92      0.76      3576
weighted avg       0.97      0.94      0.95      3576
```

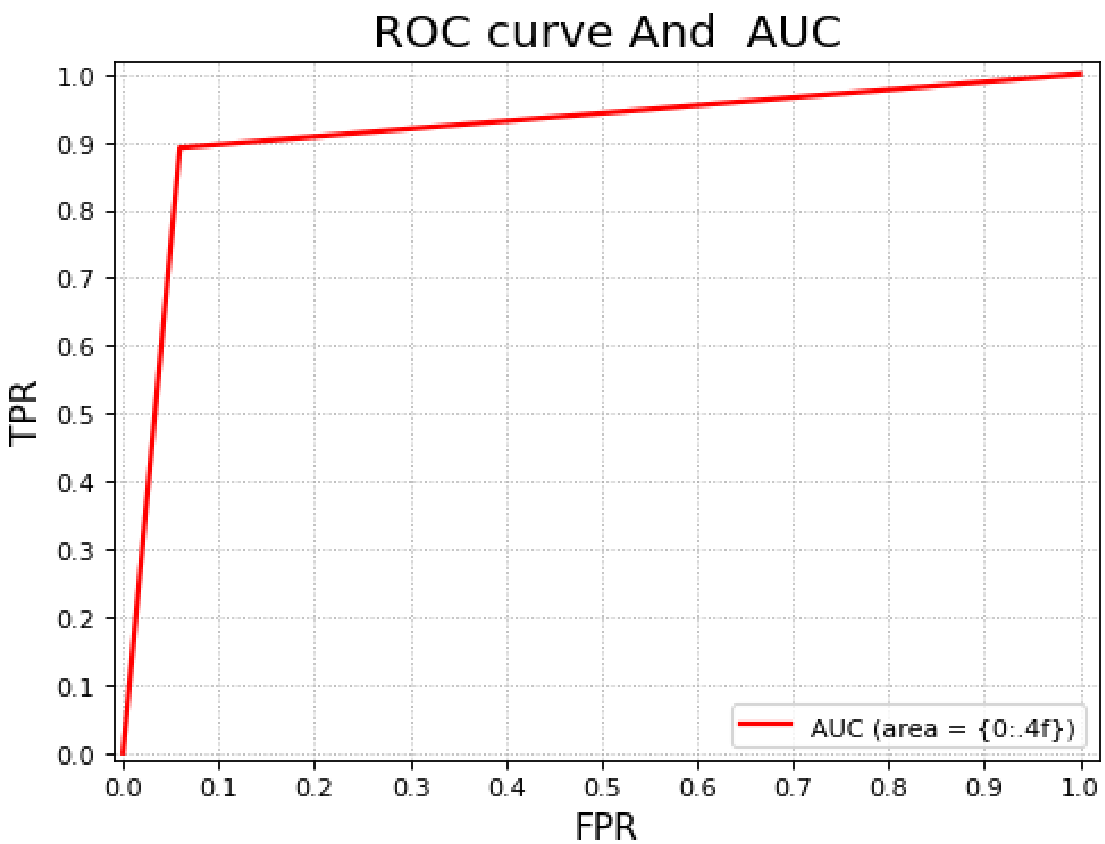
1.0.2 AUC-ROC

The results showing that we got better scores with the smaller K value. We use K=5 for the AUC-ROC graph generation, because a large number of neighbors will have a smoother decision boundary and K=5 also has accuracy greater than 90%.

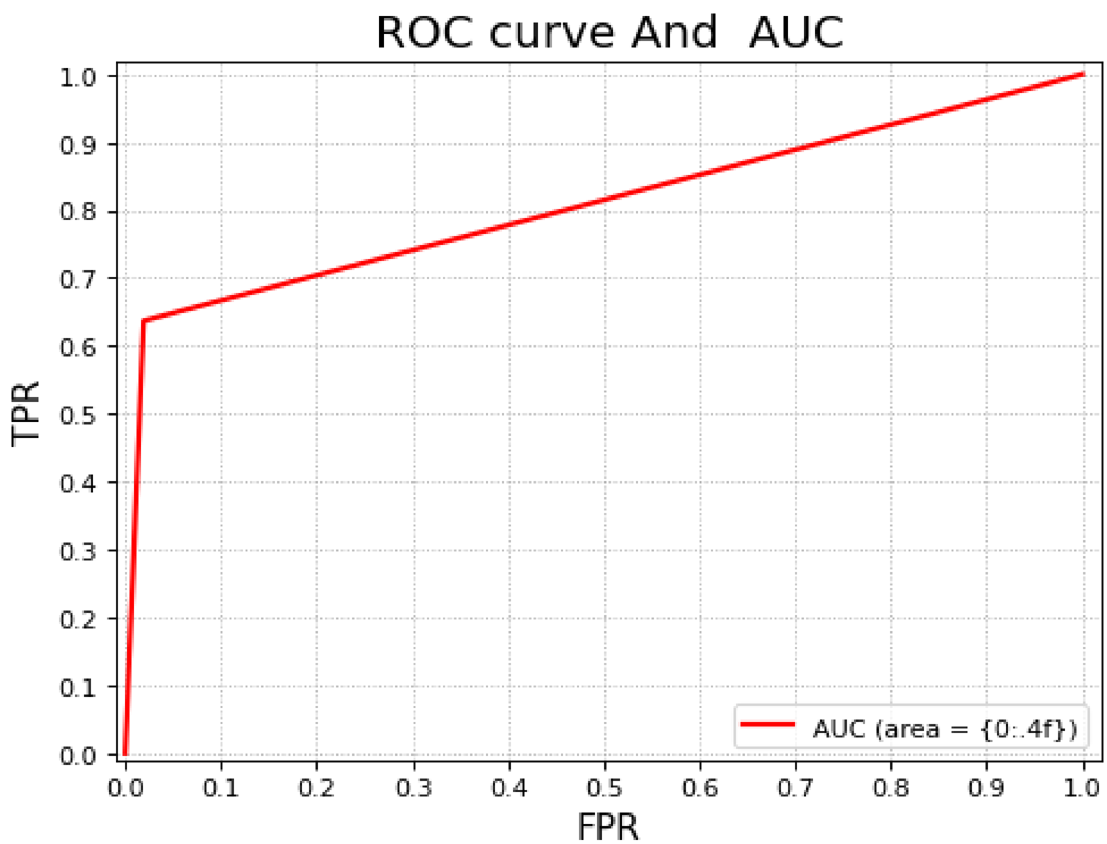
[19]:
```python
# for K=5
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred5.ravel())
auc = metrics.roc_auc_score(y_test, y_pred5, average='macro')
roc_auc = metrics.auc(fpr, tpr)
print ('AUC:\t', auc)
print ('ROC:\t', roc_auc)
```

```
AUC:	 0.9161728296164372
ROC:	 0.9161728296164372
```

[20]:
```python
# plot the ROC curve and AUC
plt.figure(figsize=(7, 5), dpi=80, facecolor='w')
plt.xlim((-0.01, 1.02))
plt.ylim((-0.01, 1.02))
plt.xticks(np.arange(0, 1.1, 0.1))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.plot(fpr, tpr, 'r-', lw=2, label='AUC (area = {0:.4f})' % auc)
plt.legend(loc='lower right')
plt.xlabel('FPR', fontsize=14)
plt.ylabel('TPR', fontsize=14)
plt.grid(visible=True, ls=':')
plt.title(u'ROC curve And  AUC', fontsize=18)
plt.show()
```



[21]:
```
# ROC curve and AUC for BoW, the unprocessed data
fpr, tpr, _ = metrics.roc_curve(y_test_unprocessed, y_pred_unprocessed.ravel())
auc = metrics.roc_auc_score(y_test, y_pred5, average='macro')

# plot the ROC curve and AUC
plt.figure(figsize=(7, 5), dpi=80, facecolor='w')
plt.xlim((-0.01, 1.02))
plt.ylim((-0.01, 1.02))
plt.xticks(np.arange(0, 1.1, 0.1))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.plot(fpr, tpr, 'r-', lw=2, label='AUC (area = {0:.4f})' % auc)
plt.legend(loc='lower right')
plt.xlabel('FPR', fontsize=14)
plt.ylabel('TPR', fontsize=14)
plt.grid(visible=True, ls=':')
plt.title(u'ROC curve And  AUC', fontsize=18)
plt.show()
```


ROC curve And AUC
TPR
FPR
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
AUC (area = {0:.4f})

```
[ ]:
```