Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

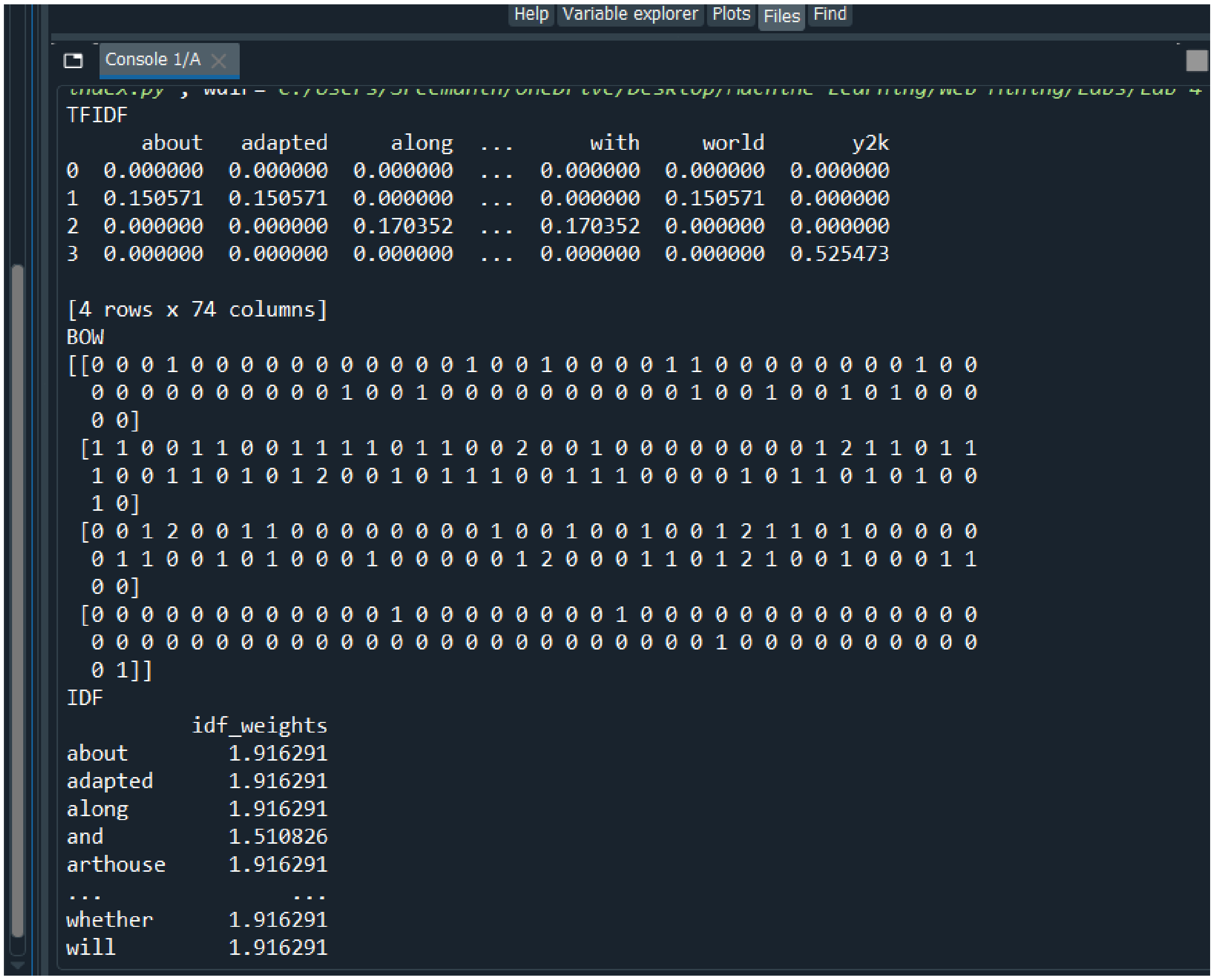
Register Number: 18BCE0745

Name: Gourishetty Sreemanth

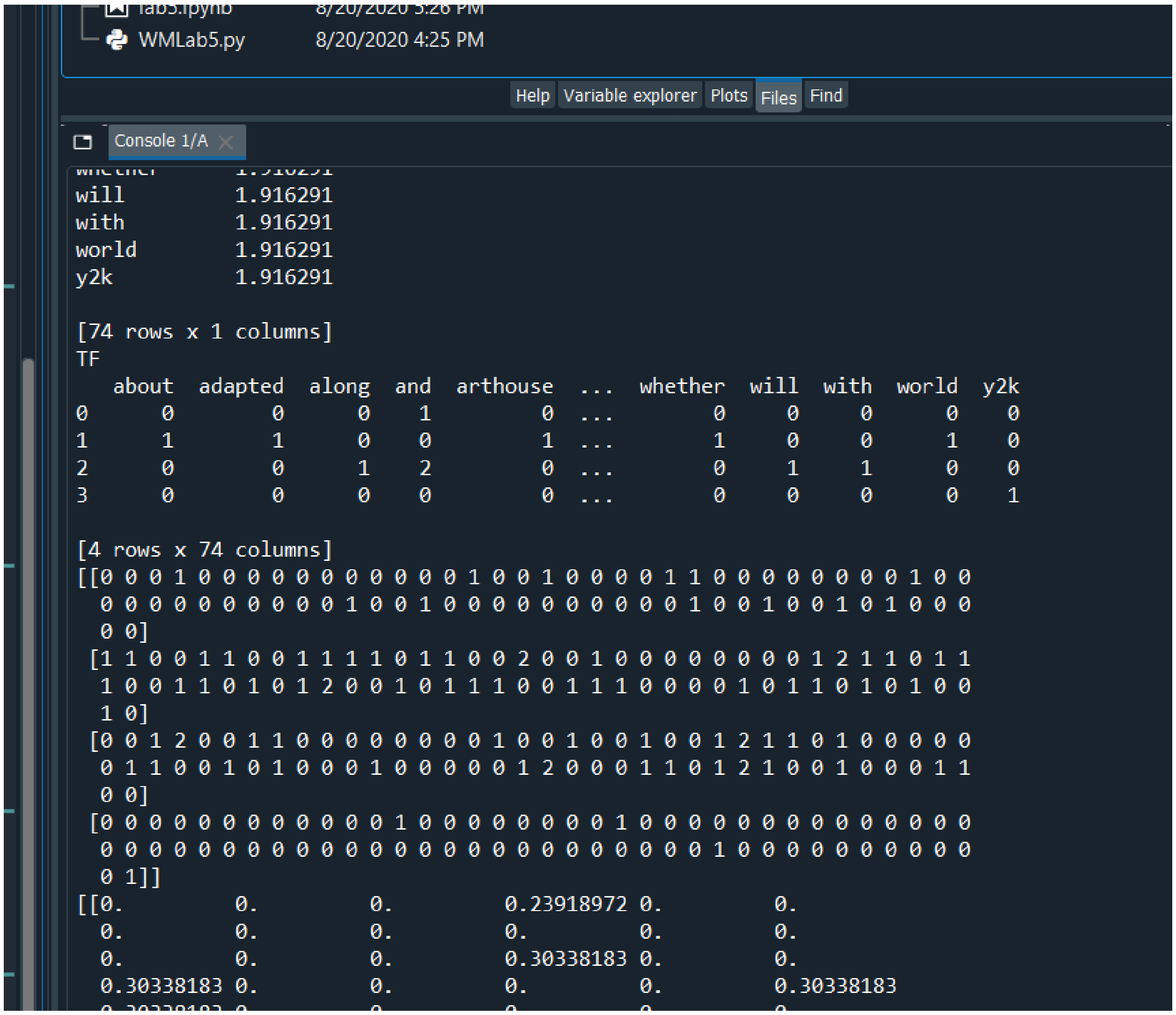
Code:

```
import pandas as pd

from sklearn.feature_extraction.text import TfidfTransformer

from sklearn.feature_extraction.text import CountVectorizer


docs=["plot: two teen couples go to a church party, drink and then drive.",

"films adapted from comic books have had plenty of success , whether they're about superheroes ( batman , superman , spawn ) , or geared toward kids ( casper ) or the arthouse crowd ( ghost world ) , but there's never really been a comic book like from hell before .",

"every now and then a movie comes along from a suspect studio , with every indication that it will be a stinker , and to everybody's surprise ( perhaps even the studio ) the film becomes a critical darling . ",

"damn that y2k bug ."

]



##########################
```

```
print("TFIDF")

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(docs)
feature_names = vectorizer.get_feature_names()
dense = vectors.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist, columns=feature_names)
print(df)
print("BOW")
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer()
word_count_vector=cv.fit_transform(docs)
word_count_vector.shape
print(word_count_vector.toarray())

print("IDF")
tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_transformer.fit(word_count_vector)

# print idf values
df_idf = pd.DataFrame(tfidf_transformer.idf_, index=cv.get_feature_names(),columns=["idf_weights"])



# sort ascending
df_idf.sort_values(by=['idf_weights'])
print(df_idf)
print("TF")

#instantiate CountVectorizer()

cv=CountVectorizer()


# this steps generates word counts for the words in your docs

word_count_vector=cv.fit_transform(docs)


word_count_vector.shape

pd = pd.DataFrame(word_count_vector.toarray(),  columns = cv.get_feature_names())

print(pd)



count_vector=cv.transform(docs)

print(count_vector.toarray())


tf_idf_vector=tfidf_transformer.transform(count_vector)

print(tf_idf_vector.toarray())


feature_names = cv.get_feature_names()


print(feature_names)
```



```
index.py', wdir='C:/Users/Sreemanth/OneDrive/Desktop/Machine Learning/Web Mining/Labs/Lab 4
TFIDF
      about   adapted     along  ...      with     world       y2k
0  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
1  0.150571  0.150571  0.000000  ...  0.000000  0.150571  0.000000
2  0.000000  0.000000  0.170352  ...  0.170352  0.000000  0.000000
3  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.525473

[4 rows x 74 columns]
BOW
[[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0
  0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0
  0 0]
 [1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 2 0 0 1 0 0 0 0 0 0 0 0 1 2 1 1 0 1 1
  1 0 0 1 1 0 1 0 1 2 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0
  1 0]
 [0 0 1 2 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 2 1 1 0 1 0 0 0 0 0
  0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1 2 0 0 0 1 1 0 1 2 1 0 0 1 0 0 0 1 1
  0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
  0 1]]
IDF
          idf_weights
about        1.916291
adapted      1.916291
along        1.916291
and          1.510826
arthouse     1.916291
...               ...
whether      1.916291
will         1.916291
```



```
will         1.916291
with         1.916291
world        1.916291
y2k          1.916291

[74 rows x 1 columns]
TF
   about  adapted  along  and  arthouse  ...  whether  will  with  world  y2k
0      0        0      0    1         0  ...        0     0     0      0    0
1      1        1      0    0         1  ...        1     0     0      1    0
2      0        0      1    2         0  ...        0     1     1      0    0
3      0        0      0    0         0  ...        0     0     0      0    1

[4 rows x 74 columns]
[[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0
  0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0
  0 0]
 [1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 2 0 0 1 0 0 0 0 0 0 0 0 1 2 1 1 0 1 1
  1 0 0 1 1 0 1 0 1 2 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0
  1 0]
 [0 0 1 2 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 2 1 1 0 1 0 0 0 0 0
  0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1 2 0 0 0 1 1 0 1 2 1 0 0 1 0 0 0 1 1
  0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 1]]
[[0.         0.         0.         0.23918972 0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.30338183 0.         0.
  0.30338183 0.         0.         0.         0.         0.30338183
```