Assignment by

FA18-BSE-010 Arose Niazi

FA18-BSE-020 Ayesha Mubashar

Importing Packages

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import pickle
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

%matplotlib inline
```

Train Dataset

```
train = pd.read_csv('train.csv', encoding = "latin", names=["Comment", "Polarity"])
train_original=train.copy()
train.shape
```

```
(10, 2)
```

```
train_original
```

	Comment	Polarity
0	@user if they want reflection money. #ksleg	Happy
1	Good job but I□ will expect a lot more in future.	Happy
2	time to eat with my best buddy! #lunch	Happy
3	totally dissatisfied with the service###%%@@ n...	Sad

Loading and making copy of test

```
test = pd.read_csv('test.csv', encoding = "latin", names=["Comment", "Polarity"])
test_original=test.copy()
test.shape
```

```
(5, 2)
```

```
test_original
```

	Comment	Polarity
0	@user the pic says otherwise for young girls c...	Sad
1	#good night! ?? #faith ever #vaitacacommafiasdv	Happy
2	@user when you're blocked by a troll because y...	Sad
3	dinner with sister!!	Happy
4	who else is planning on watching @user tomorrow?	Happy

Combining

```
combine = train.append(test,ignore_index=True,sort=True)
combine.head()
```

	Comment	Polarity
0	@user if they want reflection money. #ksleg	Happy
1	Good job but I□ will expect a lot more in future.	Happy
2	time to eat with my best buddy! #lunch	Happy
3	totally dissatisfied with the service###%%@@ n...	Sad
4	loved my work!!!!!	Happy

```
combine.tail()
```

	Comment	Polarity
10	@user the pic says otherwise for young girls c...	Sad
11	#good night! ?? #faith ever #vaitacacommafiasdv	Happy
12	@user when you're blocked by a troll because y...	Sad
13	dinner with sister!!	Happy
14	who else is planning on watching @user tomorrow?	Happy

```
def remove_pattern(text,pattern):

    # re.findall() finds the pattern i.e @user and puts it in a list for further task
    r = re.findall(pattern,text)

    # re.sub() removes @user from the sentences in the dataset
    for i in r:
        text = re.sub(i,"",text)

    return text
```

Removing @user

```
combine['Tidy_Comments'] = np.vectorize(remove_pattern)(combine['Comment'], "@[\w]*")

combine.head()
```

	Comment	Polarity	Tidy_Comments
0	@user if they want reflection money. #ksleg	Happy	if they want reflection money. #ksleg
1	Good job but I□ will expect a lot more in future.	Happy	Good job but I□ will expect a lot more in future.
2	time to eat with my best buddy! #lunch	Happy	time to eat with my best buddy! #lunch
3	totally dissatisfied with the service###%%@@ n...	Sad	totally dissatisfied with the service###%% nev...

Removing Punctuations, Numbers, and Special Characters

```
## Removing Punctuations, Numbers, and Special Characters
combine['Tidy_Comments'] = combine['Tidy_Comments'].str.replace("[^a-zA-Z#]", " ")
combine.head(10)
```

	Comment	Polarity	Tidy_Comments
0	@user if they want reflection money. #ksleg	Happy	if they want reflection money #ksleg
1	Good job but I□ will expect a lot more in future.	Happy	Good job but I will expect a lot more in future
2	time to eat with my best buddy! #lunch	Happy	time to eat with my best buddy #lunch
3	totally dissatisfied with the service###%%@@ n...	Sad	totally dissatisfied with the service### nev...
4	loved my work!!!!!	Happy	loved my work
5	Worst customer care service.....@@$$$angry	Sad	Worst customer care service angry
6	Brilliant effort guys!!!	Happy	Brilliant effort guys
7	@user you point one finger user millions are p...	Sad	you point one finger user millions are pointe...

Removing Short Words

```
combine['Tidy_Comments'] = combine['Tidy_Comments'].apply(lambda x: ' '.join([w for w in x.sp

combine.head(10)
```

	Comment	Polarity	Tidy_Comments
0	@user if they want reflection money. #ksleg	Happy	they want reflection money #ksleg
1	Good job but I□ will expect a lot more in future.	Happy	Good will expect more future
2	time to eat with my best buddy! #lunch	Happy	time with best buddy #lunch
3	totally dissatisfied with the service###%%@@ n...	Sad	totally dissatisfied with service### never use...
4	loved my work!!!!!	Happy	loved work
5	Worst customer care service.....@@$$$angry	Sad	Worst customer care service angry
6	Brilliant effort guys!!!	Happy	Brilliant effort guys
7	@user you point one finger user millions are p...	Sad	point finger user millions pointed right back ...

Tokenization

```
tokenized_text = combine['Tidy_Comments'].apply(lambda x: x.split())
tokenized_text.head()
```

```
0              [they, want, reflection, money, #ksleg]
1                   [Good, will, expect, more, future]
2                    [time, with, best, buddy, #lunch]
3    [totally, dissatisfied, with, service###, neve...
4                                          [loved, work]
Name: Tidy_Comments, dtype: object
```

Stemming

```
from nltk import PorterStemmer

ps = PorterStemmer()

tokenized_text = tokenized_text.apply(lambda x: [ps.stem(i) for i in x])

tokenized_text.head()
```

```
0                 [they, want, reflect, money, #ksleg]
1                    [good, will, expect, more, futur]
2                    [time, with, best, buddi, #lunch]
3    [total, dissatisfi, with, service###, never, u...
4                                          [love, work]
Name: Tidy_Comments, dtype: object
```

Stitching these tokens back together

```
for i in range(len(tokenized_text)):
    tokenized_text[i] = ' '.join(tokenized_text[i])

combine['Tidy_Comments'] = tokenized_text
combine.head()
```

	Comment	Polarity	Tidy_Comments
0	@user if they want reflection money. #ksleg	Happy	they want reflect money #ksleg
1	Good job but I□ will expect a lot more in future.	Happy	good will expect more futur
2	time to eat with my best buddy! #lunch	Happy	time with best buddi #lunch
3	totally dissatisfied with the service###%%@@n...	Sad	total dissatisfi with service### never use thi...

Label Encoding

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
train.apply(label_encoder.fit_transform)
```

	Comment	Polarity
0	0	0
1	3	0
2	6	0
3	7	1
4	5	0
5	4	1
6	2	0
7	1	1
8	8	0
9	9	1

```
label_encoder2 = preprocessing.LabelEncoder()
test.apply(label_encoder2.fit_transform)
```

	Comment	Polarity
0	1	1
1	0	0
2	2	1
3	3	0
4	4	0

Extracting Features from cleaned Tweets

Bag-of-Words Features

```
from sklearn.feature_extraction.text import CountVectori
```

Using the features from TF-IDF

```
train_tfidf_matrix = tfidf_matrix[:10]

train_tfidf_matrix.todense()
```

```
matrix([[0.        , 0.        , 0.        , 0.        ],
        [1.        , 0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        ],
        [0.        , 0.        , 1.        , 0.        ],
        [0.        , 1.        , 0.        , 0.        ],
        [0.        , 0.        , 1.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 1.        ],
        [0.        , 0.70710678, 0.        , 0.70710678],
        [0.        , 0.        , 0.        , 0.        ]])
```

```
from sklearn.model_selection import train_test_split
x_train_bow,x_valid_bow,y_train_bow,y_valid_bow = train_test_split(train_bow,train['Polarity'
x_train_tfidf,x_valid_tfidf,y_train_tfidf,y_valid_tfidf = train_test_split(train_tfidf_matrix
```

Decision Tree

Using Bag of words feature

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
dct = DecisionTreeClassifier(criterion='entropy', random_state=1)


dct.fit(x_train_bow,y_train_bow)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1, splitter='best')
```

```
dct_bow = dct.predict_proba(x_valid_bow)

dct_bow
```

```
from sklearn.feature_extraction.text import CountVectorizer

bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='englis

# bag-of-words feature matrix
bow = bow_vectorizer.fit_transform(combine['Tidy_Comments'])

df_bow = pd.DataFrame(bow.todense())

df_bow
```

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	0	0	0	0
3	0	0	1	0
4	0	1	0	0
5	0	0	1	0
6	0	0	0	0
7	0	0	0	1
8	0	1	0	1
9	0	0	0	0
10	0	0	0	0
11	1	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0

TF-IDF Features

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf=TfidfVectorizer(max_df=0.90, min_df=2,max_features=1000,stop_words='english')

tfidf_matrix=tfidf.fit_transform(combine['Tidy_Comments'])

df_tfidf = pd.DataFrame(tfidf_matrix.todense())

df_tfidf
```

```
array([[1.  , 0.  ],
       [0.75, 0.25],
       [0.  , 1.  ]])
```

if prediction is greater than or equal to 0.3 than 1 else 0

Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets New Section

converting the results to integer type

calculating f1 score

```
y_int_valid_bow=y_valid_bow.replace('Happy',0)
y_int_valid_bow=y_int_valid_bow.replace('Sad',1)


dct_bow=dct_bow[:,1]>=0.3

dct_int_bow=dct_bow.astype(np.int)

dct_score_bow=f1_score(y_int_valid_bow,dct_int_bow)

dct_score_bow
```

```
1.0
```

Using TF-IDF Features

```
dct.fit(x_train_tfidf,y_train_tfidf)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1, splitter='best')
```

```
dct_tfidf = dct.predict_proba(x_valid_tfidf)

dct_tfidf
```

```
array([[0.66666667, 0.33333333],
       [0.66666667, 0.33333333],
       [0.        , 1.        ]])
```

```
y_int_valid_tfidf=y_valid_tfidf.replace('Happy',0)
y_int_valid_tfidf=y_int_valid_tfidf.replace('Sad',1)
```

	0	1	2	3
0	0.0	0.000000	0.0	0.000000
1	1.0	0.000000	0.0	0.000000
2	0.0	0.000000	0.0	0.000000
3	0.0	0.000000	1.0	0.000000
4	0.0	1.000000	0.0	0.000000
5	0.0	0.000000	1.0	0.000000
6	0.0	0.000000	0.0	0.000000
7	0.0	0.000000	0.0	1.000000
8	0.0	0.707107	0.0	0.707107
9	0.0	0.000000	0.0	0.000000
10	0.0	0.000000	0.0	0.000000
11	1.0	0.000000	0.0	0.000000
12	0.0	0.000000	0.0	0.000000
13	0.0	0.000000	0.0	0.000000
14	0.0	0.000000	0.0	0.000000

MACHINE LEARNING

First using the features from Bag-of-Words

```
train_bow = bow[:10]

train_bow.todense()
```

```
matrix([[0, 0, 0, 0],
        [1, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 1, 0],
        [0, 1, 0, 0],
        [0, 0, 1, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 1],
        [0, 1, 0, 1],
        [0, 0, 0, 0]])
```

```
y_int_valid_tfidf=y_int_valid_tfidf.replace('Sad',1)

dct_tfidf=dct_tfidf[:,1]>=0.3


dct_int_tfidf=dct_tfidf.astype(np.int)

dct_score_tfidf=f1_score(y_int_valid_tfidf,dct_int_tfidf)

dct_score_tfidf
```

```
0.8
```

Using Decision Tree Model to Predict for the Test Data

```
test_tfidf = tfidf_matrix[10:]

test_pred = dct.predict_proba(test_tfidf)

test_pred_int = test_pred[:,1] >= 0.3

test_pred_int = test_pred_int.astype(np.int)

test['Polarity'] = test_pred_int

submission = test[['Comment','Polarity']]

submission.to_csv('result.csv', index=False)
```

```
res = pd.read_csv('result.csv')
res
```

	Comment	Polarity
0	@user the pic says otherwise for young girls c...	1
1	#good night! ?? #faith ever #vaitacacommafiasdv	0
2	@user when you're blocked by a troll because y...	1
3	dinner with sister!!	1
4	who else is planning on watching @user tomorrow?	1

```
res['Polarity'] = res['Polarity'].replace([1],'Sad')
res['Polarity'] = res['Polarity'].replace([0],'Happy')
res
```

	Comment	Polarity
0	@user the pic says otherwise for young girls c...	Sad
1	#good night! ?? #faith ever #vaitacacommafiasdv	Happy
2	@user when you're blocked by a troll because y...	Sad
3	dinner with sister!!	Sad
4	who else is planning on watching @user tomorrow?	Sad

```
combine = train.append(res,ignore_index=True,sort=True)
combine
```

	Comment	Polarity
0	@user if they want reflection money. #ksleg	Happy
1	Good job but I□ will expect a lot more in future.	Happy
2	time to eat with my best buddy! #lunch	Happy
3	totally dissatisfied with the service###%%@@ n...	Sad
4	loved my work!!!!!	Happy
5	Worst customer care service.....@@$$$angry	Sad
6	Brilliant effort guys!!!	Happy
7	@user you point one finger user millions are p...	Sad
8	words r free it's how u use that can cost you!...	Happy
9	you might be a libtard if... #libtard #sjw #li...	Sad
10	@user the pic says otherwise for young girls c...	Sad
11	#good night! ?? #faith ever #vaitacacommafiasdv	Happy
12	@user when you're blocked by a troll because y...	Sad
13	dinner with sister!!	Sad
14	who else is planning on watching @user tomorrow?	Sad

Save the Trained Model

```
pickle.dump(dct, open('trainedModel.pkl', 'wb'))
```

Take Input from User

```
Sentence_input = input("\nPlease enter a sentence (Something positive/negative) : ").strip()
```

```
Please enter a sentence (Something positive/negative) : ayesha is my group member arose
```

Convert User Input into Feature Vector

```
user_input = pd.DataFrame({ 'id': ['1'],'Comment': [Sentence_input]})

print(user_input)
```

```
User Input Feature Vector:
==========================

  id                          Comment
0  1  ayesha is my group member arose
```

Reapply everything to make it useable

```
user_input['Tidy_Comments'] = np.vectorize(remove_pattern)(user_input['Comment'], "@[\w]*")
user_input['Tidy_Comments'] = user_input['Tidy_Comments'].str.replace("[^a-zA-Z#]", " ")
user_input['Tidy_Comments'] = user_input['Tidy_Comments'].apply(lambda x: ' '.join([w for w i
tokenized_comment = user_input['Tidy_Comments'].apply(lambda x: x.split())

ps = PorterStemmer()
tokenized_comment = tokenized_comment.apply(lambda x: [ps.stem(i) for i in x])

for i in range(len(tokenized_comment)):
    tokenized_comment[i] = ' '.join(tokenized_comment[i])

user_input['Tidy_Comments'] = tokenized_comment

user_input['Tidy_Comments']
```

```
0    ayesha group member aros
Name: Tidy_Comments, dtype: object
```

extracting features

```
bow_vectorizer = CountVectorizer(max_df=1.0, min_df=1, max_features=10, stop_words='english')

bow = bow_vectorizer.fit_transform(user_input['Tidy_Comments'])
df_user_bow = pd.DataFrame(bow.todense())

df_user_bow
```

	0	1	2	3
0	1	1	1	1

extracting features

```
tfidf=TfidfVectorizer(max_df=1.0, min_df=1,max_features=1000,stop_words='english')

# TF-IDF feature matrix
user_tfidf_matrix=tfidf.fit_transform(user_input['Tidy_Comments'] )
# convert to data frame
df_user_tfidf = pd.DataFrame(user_tfidf_matrix.toarray())

df_user_tfidf
```

	0	1	2	3
0	0.5	0.5	0.5	0.5

Load the Saved Model

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

```
input_pred = model.predict_proba(user_tfidf_matrix)

test_pred_int = test_pred[:,1] >= 0.3

test_pred_int = test_pred_int.astype(np.int)
```

```
print(test_pred_int )
```

```
     [1 0 1 1 1]
```