DATA SCIENCE WITH PYTHON

CS 498 (SPECIAL TOPICS IN COMPUTER SCIENCE)

LAB FOR MACHINE LEARNING

Dr. Malak Abdullah

Computer Science Department

Jordan University of Science and Technology

FOLLOW THE INSTRUCTION

Your grade will be as much as you can finish

لعند ما بتوصل بالمختبر بتاخد علامتك

☺ اشتغل اكثر بتاخد اكثر

READ DATA

- This is a supervised text classification practice
- At the beginning read the data from the file anger.txt .(on e-learning)

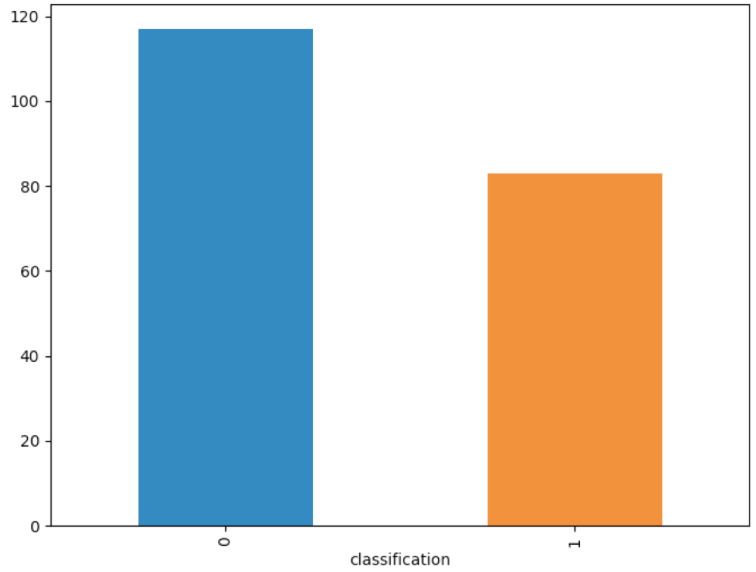
```
import pandas as pd
data = pd.read_csv('anger.txt', header=None, delimiter="\t")
print (data.shape)

data.columns= ["id", "tweet", "emotion", "classification"]
```

EXPLORE THE DATA

- Take a look on the file by printing the head and the tail
- Also, see if the data are balanced. Its very important to have a balanced data!!
- You can do this by plotting the data

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
data.groupby('classification').id.count().plot.bar(ylim=0)
plt.show()
```



CLEAN THE TWEETS

```
import re
#start process_tweet
def processTweet(tweet):
    # process the tweets

    #Convert to lower case
    tweet = tweet.lower()
    #Convert www.* or https?://* to URL
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
    #Convert @username to AT_USER
    tweet = re.sub('@[^\s]+','AT_USER',tweet)
    #Remove additional white spaces
    tweet = re.sub('[\s]+', ' ', tweet)
    #Replace #word with word
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
    #trim
    tweet = tweet.strip('\'"')
    return tweet
#end

for i in range(200):
    data.tweet[i]=processTweet(data.tweet[i])
print (data.tweet.head())
```

- You can use your own or use the code in this screenshot
- You don't have to finish all of them!!

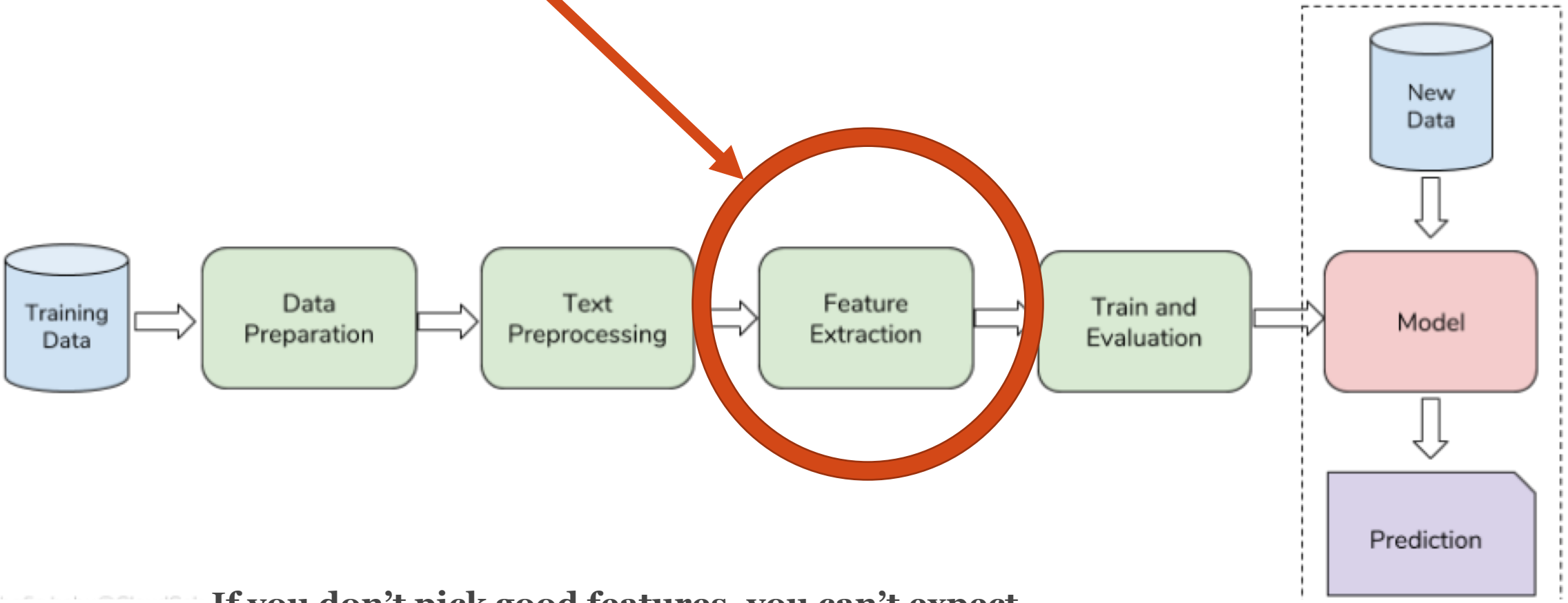
SPLIT THE DATA

- Split the data into train %75 and test %25

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.tweet, data.classification, test_size=0.25)
```

YOU ARE HERE

You need to extract the feature of each tweet



If you don't pick good features, you can't expect your model to work well

EXTRACTING FEATURES USING NLP TECHNIQUES

- **Term frequency** is simply the ratio of the count of a word present in a sentence, to the length of the sentence. Therefore, we can generalize term frequency as:

TF = (Number of times term T appears in the particular row) / (number of terms in that row)

- **inverse document frequency (IDF):** Think about IDF as a measure of uniqueness. A word is not of much use to us if it's appearing in all the documents.
- The IDF of each word is the log of the ratio of the total number of rows to the number of rows in which that word is present.

IDF = log(N/n), where, N is the total number of rows and n is the number of rows in which the word was present.

TF-IDF

- TF-IDF are word frequency scores that try to highlight words that are more interesting, e.g. frequent in a document but not across documents.
- The TfidfVectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.

- We will create an encoded vector for each tweet ☺ by using TfidfVectorizer()
- Call the *fit()* function in order to learn a vocabulary from one or more documents.
- Call the *transform()* function on one or more documents as needed to encode each as a vector.

THIS IS TWEET 9 ☺

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
#vectorizer = TfidfVectorizer(min_df=5, max_df = 0.8,sublinear_tf=True,
#                              use_idf=True,decode_error='ignore')    Play with this :)
vectorizer = TfidfVectorizer()
train_vectors = vectorizer.fit_transform(X_train)
test_vectors = vectorizer.transform(X_test)
```

```
  (0, 59)   0.180010585734825 67
  (0, 48)   0.5825851127145155
  (0, 62)   0.294726509877089 26
  (0, 11)   0.294726509877089 26
  (0, 23)   0.3110671259851867
  (0, 24)   0.344084152519952 34
  (0, 68)   0.269470591937623 96
  (0, 53)   0.238283530209056 65
  (0, 46)   0.3315316616041348
is it supposed to be this arduous to interact with other people? i'm 'other people' to them and they don't seem to struggle.
```

NOW THE CLASSIFICATION

- We will use SVM ☺

```
from sklearn import svm
from sklearn.svm import SVC
# Perform classification with SVM, kernel=linear
clf = svm.SVC(kernel='linear')
clf.fit(train_vectors, y_train)
y_pred = clf.predict(test_vectors)
```

EVALUATE THE RESULTS

```
61
62  from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
63
64  print (confusion_matrix(y_test, y_pred))
65  print(classification_report(y_test, y_pred))
66  print('\nAccuracy: {0:.4f}'.format(accuracy_score(y_test, y_pred)))
67
```

```
[[23  7]
 [14  6]]
             precision    recall  f1-score   support

          0       0.62      0.77      0.69        30
          1       0.46      0.30      0.36        20

avg / total       0.56      0.58      0.56        50


Accuracy: 0.5800
```

THE LAB IS DONE, GOOD JOB!!! ☺

- Submit your code and a print screen of your code and the results!!

ASSIGNMENT

- I didn't like the results!!
- Try to do naïve bays and decision tree to find out if the results are better than this.
- Submit the code
- Submit a report with print screens of the results to compare the three results
- Due-Date 1/12/2018 at 23:55

- Good luck ☺