Text Classification

Text classification is the process of categorizing text into organized groups. It can be applied to words, sentences, or entire documents. The goal is to automatically understand the content of the text and sort it into the correct category based on its meaning or context.

- Reading Data
- handle dataset
- Text Cleaning
- TF-IDF Vecorization
- Feature Creation
- ML Classifiers

Reading Data

In [186]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
# to increasing col width
pd.set_option('display.max_colwidth',100)
dataset = pd.read_csv('spam.csv')
dataset.head()
```

Out[186]:

	label	text
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives around here though

handle dataset

In [193]:
```
dataset["label"] = dataset["label"].replace({'ham': 0, 'spam': 1})
dataset.head()
```

Out[193]:

	label	text
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives around here though

NLP PIPELINE

Row Text -> Tokenization -> Text Cleaning -> Vectorization -> ML Algorithm -> classifying text

Preprocessing = Tokenization and Text Cleaning

vectorization = Convert text to numbers

vectorization methods (word2vec - BOW - TFIDF)

Preprocessing

Preprocessing (Removing Punctuation - Tokenization - Remove Stop Words - stemming/lemmatizing)

Lemmatization

lemmatization is more accurate but computationally expensive

lemmatization reduces to a dictionary word

Vectorization

Vectorization : process of encoding text as integers to create feature vecors

Feature vector : vector of numerical features that represent an object

Types Of Vecorization (count vectorization - Ngrams - TFIDF)

count vectorization +> count unique words occure in the sms how many times!!

Text Cleaning

apply on our dataset

In [195]:
```
import string
from nltk import word_tokenize
import nltk
from nltk.stem import PorterStemmer
ps = PorterStemmer()
#nltk.download('punkt')
#nltk.download('stopwords.words('english')
def clean_text(txt):
    txt_nopunct = "".join([c for c in txt if c not in string.punctuation ])
    tokens = word_tokenize(txt_nopunct)
    txt_clean = [word for word in tokens if word not in stopwords]
    tokens_stem = [ps.stem(word) for word in txt_clean]
    #tokens_lemma = [wn.lemmatize(word) for word in txt_clean]
    return tokens_stem
```

TF-IDF Vecorization

In [208]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# cv2 = CountVectorizer(analyzer=clean_text)# if i want to perform cleaning before it
tfidf_vec = TfidfVectorizer(analyzer=clean_text)
tfidf_vec_fit = tfidf_vec.fit(dataset["text"])
X_tfidf = tfidf_vec.fit_transform(dataset["text"])
print(X_tfidf.shape)
#pd.DataFrame(X_tfidf.toarray(),columns=tfidf_vec.get_feature_names_out())
#df.head()
```

(5572, 8176)

Feature Engineering

- creating new features of transforming existing features using domain knowledge of the data, that make machine learning

algorithm work better

- creating features
  - length of documents
  - average word size within a document
  - use of punctuation in the text
  - capitalization of words in a document
  - ...
  - ...
  - ...
- Transformations(applying some transformations to data can make it work better)
  - Power transformations (x^2 , √x ,etc) #if = all251
  - Standardizing data
  - Normalization : bring different features to similar scale

Feature Creation

message length - punctuation usage - stop word usage - capitalization usage - average word length usage ....

message length

In [213]:
```
dataset["text_len"] = dataset["text"].apply(lambda x: len(x))
dataset.head()
```

Out[213]:

	label	text	text_len
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives around here though	61

punctuation length

In [214]:
```
import string
def punctuation_count(txt):
    count= sum([1 for c in txt if c in string.punctuation])
    return (count / len(txt)) * 100
```

In [215]:
```
dataset["punctuation_%"] = dataset["text"].apply(lambda x: punctuation_count(x))
dataset.head()
```

Out[215]:

	label	text	text_len	punctuation_%
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g...	111	8.108108
1	0	Ok lar... Joking wif u oni...	29	20.689655
2	1	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ...	155	3.870968
3	0	U dun say so early hor... U c already then say...	49	12.244898
4	0	Nah I don't think he goes to usf, he lives around here though	61	3.278689

ML Classifiers

In [220]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import precision_score, recall_score, accuracy_score,confusion_matrix,classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
```

In [221]:
```
#dataset['hate'] = dataset['label'].map( {'off': 1, 'notOff': 0} ).astype(int)
dataset.head(5)
```

Out[221]:

	label	text	text_len	punctuation_%
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there g...	111	8.108108
1	0	Ok lar... Joking wif u oni...	29	20.689655
2	1	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ...	155	3.870968
3	0	U dun say so early hor... U c already then say...	49	12.244898
4	0	Nah I don't think he goes to usf, he lives around here though	61	3.278689

In [224]:
```
from sklearn.model_selection import train_test_split

data_tfidf_train, data_tfidf_test, label_train, label_test = train_test_split(X_tfidf, dataset["label"], test_size=0.3, random_state=42)
```

1-Naive Bayes Classifier

In [227]:
```
spam_detect_model = MultinomialNB().fit(data_tfidf_train, label_train)
pred_test_MNB = spam_detect_model.predict(data_tfidf_test)
precision = precision_score(label_test, pred_test_MNB)
recall = recall_score(label_test, pred_test_MNB)
accuracy = accuracy_score(label_test, pred_test_MNB)
print('Precision: {} / Recall: {} / Accuracy: {}'.format(round(precision, 3), round(recall, 3), round(accuracy, 3)))
print(confusion_matrix(label_test,pred_test_MNB))
print (classification_report(label_test, pred_test_MNB))
```

```
Precision: 1.0 / Recall: 0.653 / Accuracy: 0.955
[[1453   0]
 [  76  143]]
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      1453
           1       1.00      0.65      0.79       219

    accuracy                           0.95      1672
   macro avg       0.98      0.83      0.88      1672
weighted avg       0.96      0.95      0.95      1672
```

2-Decision Tree Classifier

In [233]:
```
spam_detect_model = tree.DecisionTreeClassifier().fit(data_tfidf_train, label_train)
pred_test_MNB = spam_detect_model.predict(data_tfidf_test)
precision = precision_score(label_test, pred_test_MNB)
recall = recall_score(label_test, pred_test_MNB)
accuracy = accuracy_score(label_test, pred_test_MNB)
print('Precision: {} / Recall: {} / Accuracy: {}'.format(round(precision, 3), round(recall, 3), round(accuracy, 3)))
print(confusion_matrix(label_test,pred_test_MNB))
print (classification_report(label_test, pred_test_MNB))
```

```
Precision: 0.877 / Recall: 0.813 / Accuracy: 0.961
[[1428   25]
 [  41  178]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      1453
           1       0.88      0.81      0.84       219

    accuracy                           0.96      1672
   macro avg       0.92      0.90      0.91      1672
weighted avg       0.96      0.96      0.96      1672
```

3- Random Forest Classifier

In [236]:
```
spam_detect_model = RandomForestClassifier().fit(data_tfidf_train, label_train)
pred_test_MNB = spam_detect_model.predict(data_tfidf_test)
precision = precision_score(label_test, pred_test_MNB)
recall = recall_score(label_test, pred_test_MNB)
accuracy = accuracy_score(label_test, pred_test_MNB)
print('Precision: {} / Recall: {} / Accuracy: {}'.format(round(precision, 3), round(recall, 3), round(accuracy, 3)))
print(confusion_matrix(label_test,pred_test_MNB))
print (classification_report(label_test, pred_test_MNB))
```

```
Precision: 1.0 / Recall: 0.808 / Accuracy: 0.975
[[1453   0]
 [  42  177]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      1453
           1       1.00      0.81      0.89       219

    accuracy                           0.97      1672
   macro avg       0.99      0.90      0.94      1672
weighted avg       0.98      0.97      0.97      1672
```

test the model

In [239]:
```
text = "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat..."
X = tfidf_vec_fit.transform([text])
pred = spam_detect_model.predict(X)
if pred[0]==1:
    print('ham')
else:
    print('spam')
```

```
ham
```

In [241]:
```
text = "SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info"
X = tfidf_vec_fit.transform([text])
pred = spam_detect_model.predict(X)
print(pred)
if pred[0]==1:
    print('ham')
else:
    print('spam')
```

```
[1]
spam
```

Save Model Component

In [244]:
```
import pickle
with open('tfidf_vec_fit.pickle', 'wb') as handle:
    pickle.dump(tfidf_vec_fit, handle)

# save the model to disk
filename = 'RandomForest.sav'
pickle.dump(spam_detect_model, open(filename, 'wb'))
```

load Model Component

In [247]:
```
with open('tfidf_vec_fit.pickle', 'rb') as handle:
    tfidf_vec_fit_loaded = pickle.load(handle)

with open('RandomForest.sav', 'rb') as handle:
    spam_detect_model_loaded = pickle.load(handle)
```

predict from loaded model component

In [254]:
```
text = "Please call our customer service representative on FREEPHONE 0808 145 4742 between 9am-11pm as you have WON a guaranteed ÂŁ1000 cash or ÂŁ5000 prize!"
X = tfidf_vec_fit_loaded.transform([text])
pred = spam_detect_model_loaded.predict(X)
if pred[0]==1:
    print('ham')
else:
    print('spam')
```

```
spam
```

Evaluation Metrics

In [122]:
```
#accuracy = #predicted correctly! / #observations)
#precision = #predicted as spam correctly! / #predicted as spam)
#recall   = #predicted as spam correctly! / #actual spam)
```

In [143]:
```
# another machine learning algorithms
# LogisticRegression
# Support Vector Machine (SVM)
# KNN
# XGBClassifier Model
#...
```

In [ ]: