Project - 1 Text classification

Text classification is an application in natural language processing in which we can create a model, to classify human language into different classes

→ Ex : Spam Email.
↳ gmail is using a spam filter (Text classification)
↳ It can also be used in Sentiment Analysis

→ Sentiment classifier, spam filter we can use this dataset

```
# text classification.
import numpy as np
import re
import pickle
import nltk
from nltk.corpus import stopwords
from sklearn.datasets import load_files
nltk.download('stopwords')

# importing dataset
reviews = load_files('txt_sentoken/')
X, y = reviews.data, reviews.target

* load file will take very long time

# storing as Pickle Files
  # Pickles are byte type files
with open('X.pickle', 'wb') as f:
    pickle.dump(X, f)
```

```
with open('y.pickle', 'wb') as f:
    pickle.dump(y,f)

# Unpickling the dataset
with open('X.pickle', 'rb') as f:
    X = pickle.load(f)

with open('y.pickle', 'rb') as f:
    y = pickle.load(f)

# Using Pickle the data retrival will speed up.

# Preprocessing
    # Creating the corpus
corpus = []
for i in range(0, len(X)):
    review = re.sub(r'\W', ' ', str(X[i]))
    review = review.lower()
    review = re.sub(r'\s+[a-z]\s+', ' ', review)
    review = re.sub(r'^[a-z]\s+', ' ', review)
    review = re.sub(r'\s+', ' ', review)
    corpus.append(review)

from sklearn.feature_Extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=2000, min_df=3, max_df=0.6, stop_words=
                             stopwords.words('english'))
X = vectorizer.fit_transform(corpus).toarray()
```

Converting Bow model into TFIDF model

TFIDF Transformer has the capability to convert the Bag of words model into a TFIDF model.

```
from sklearn.feature_extraction.text import TfidfTransformer
transformer = TfidfTransformer()
X = transformer.fit_transform(X).toarray()
```

Splitting the dataset into training and testing.

```
from sklearn.model_selection import train_test_split
_____ = train_test_split(X, y, test_size=0.2, random_state=0)
```

text_train, text_test, sent_train, sent_test

Logistic Regression – Binary classification

* The sentiment analysis task is mainly a binary classification problem to predict whether a given sentence is positive or negative. In our demonstrations we denote as negative and '1' as positive.

The point concept

* Each sentence is mapped to a point
* If the point is greater than 0.5 then positive else negative.

* A learning algorithm is a specific type of algorithm whose performance increases with time. Logistic regression is a type of learning algorithm. It learns from a training dataset, the pattern of the data and applies the learned logic on new data for predictions

Linear Equation

$$y = a + bx_1 + cx_2 \cdots dx_{2000}$$

a, b, c, d: Co-efficients

$x_1, x_2 \cdots x_{20000}$ = Independent Variables

y = dependent Variable.

The algorithm find the optimal value of Co-efficients.

If $y >= 0.5$ +ve Sentiment

If $y < 0.5$ -ve sentiment

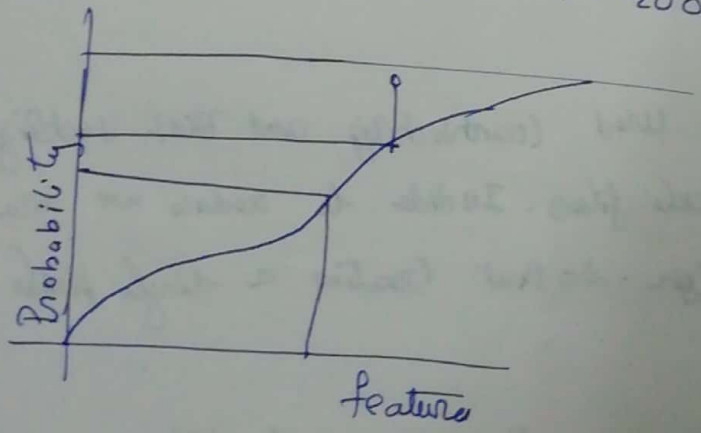
For some values of the dependent Variable, the value of Y can be >1 or <0

For that, we need some way to restrict the value of Y within the range 0 and 1

For $y > 0$ $\quad y = e^{(a + bx_1 + cx_2 + \cdots + dx_{2000})}$

$y < 1$ $\quad y = \dfrac{e^{(a + bx_1 + cx_2 + \cdots + dx_{20000})}}{e^{(a + bx_1 + cx_2 + \cdots + dx_{2000})} + 1}$

$$\ln\left(\frac{y}{y-1}\right) = a + bx_1 + cx_2 + \cdots + dx_{2000}$$



Applying Logistic Regression

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(text_train, sent_train)

Sent_pred = classifier.predict(text_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Sent_test, sent_pred)
```

	0	1	
0	168 TN	40	Fp
1	21	171	
	FN	Tp	

#339

$$\text{Accuracy} = \frac{cm[0][0] + cm[1][1]}{cm[0][0] + [0][1] + [1][0] + [1][1]}$$

Saving the classifier

```
with open('classifier.pickle', 'wb') as f:
    pickle.dump(classifier, f)
```

In the above code we have used CountVectorizer and TfidfVectorizer so we have to create two pickle files. In order to reduce we are creating a single TfidfVectorizer. So that creating a single pickle file will work.

```
from sklearn.feature_extraction.text import TfidfVectorizer
Vectorizer = TfidfVectorizer(max_features=2000, min_df=3, max_df=0.6,
    stopwords = stopwords.words('English'))

X = vectorizer.fit_transform(corpus).toarray()
```

```
# Pickling the Vectorizer
with open('tfidfmodel.pickle', 'wb') as f:
    pickle.dump(vectorizer, f)

# Importing and using our model
# unpickling the classifier and vectorizer
with open('classifier.pickle', 'rb') as f
    clf = Pickle.load(f)
with open('tfidfmodel.pickle', 'rb') as f:
    tfidf = pickle.load(f)

sample = ["you are a nice person man, have a good life"]
sample = tfidf.transform(sample).toarray()
print(clf.predict(sample))
```

How apply a user defined function on the specific column

```
df["cleare"] : df.Desc.apply(remove_noise)
```

↑ dataframe ↑ column

↑ function name

How to apply map/Lambda on a Column

```
lemmatizer = WordNetLemmatizer()
df['Stemmed'] = df.cleared.map(lambda x: ' '.join([lemmatize.lemmatize(y) for y in x]))
```