### **What is Sentiment Analysis?**

Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

In simple English:

We use computers to extract meanings behind texts, images and other data.

### Why do we need Sentiment Analysis?

Why can’t humans just read the texts? Why do we need a machine to do it for us?

Reasons for using sentiment analysis:

* Machines can read much faster (maybe a million times faster) than humans
* Machines can read in many languages
* Machines can derive meaning from text in a standardised manner (humans are subjective)
* Machines can store insights from texts in a convenient way for further processing

There are of course downsides to sentiment analysis. Machines are not able to accurately derive meaning from texts (but they are getting better). Slangs, typos, contextual meaning, sarcasm still poses difficulties.

## Technologies Used

* NLP
* Pandas
* NLTK
* Flask
* HTML
* CSS
* Bootstrap

**Code :**

|  |  |
| --- | --- |
| from flask import Flask, request, jsonify, render\_template |  |
|  | import pickle |
|  | import pandas as pd |
|  |  |
|  | app = Flask(\_\_name\_\_) |
|  |  |
|  | model = pickle.load(open('trained-model.pkl', 'rb')) |
|  |  |
|  |  |
|  | @app.route('/') |
|  | def home(): |
|  | return render\_template('home.html') |
|  |  |
|  |  |
|  | @app.route("/analyse", methods=['POST', 'GET']) |
|  | def predict(): |
|  | dataset = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter='\t', quoting=3) |
|  |  |
|  | # Cleaning the texts |
|  | import re |
|  | import nltk |
|  | nltk.download('stopwords') |
|  | from nltk.corpus import stopwords |
|  | from nltk.stem.porter import PorterStemmer |
|  | corpus = [] |
|  | for i in range(0, 1000): |
|  | review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i]) |
|  | review = review.lower() |
|  | review = review.split() |
|  | ps = PorterStemmer() |
|  | review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))] |
|  | review = ' '.join(review) |
|  | corpus.append(review) |
|  |  |
|  | # Creating the Bag of Words model |
|  | from sklearn.feature\_extraction.text import CountVectorizer |
|  | cv = CountVectorizer(max\_features=1500) |
|  | X = cv.fit\_transform(corpus).toarray() |
|  | y = dataset.iloc[:, 1].values |
|  |  |
|  | # Splitting the dataset into the Training set and Test set |
|  | from sklearn.model\_selection import train\_test\_split |
|  | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=0) |
|  |  |
|  | # Fitting Naive Bayes to the Training set |
|  | from sklearn.naive\_bayes import GaussianNB |
|  | classifier = GaussianNB() |
|  | classifier.fit(X\_train, y\_train) |
|  | classifier.score(X\_test, y\_test) |
|  |  |
|  | message = request.form['review'] |
|  | data = [message] |
|  | vector = cv.transform(data).toarray() |
|  | prediction = model.predict(vector) |
|  | return render\_template('home.html', prediction=prediction) |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | app.run(debug=True) |