# *Naive Bayes Classifier*

## Group Members:

**Sarthak Gaur - 2017A7PS0250H**
**Smit D Sheth - 2017A7PS1666H**
**Sourav Sanganeria - 2017A7PS1625H**

## Introduction:

The aim of this assignment was to build a **Naive Bayes classifier** that could perform **Sentiment Analysis** on a '*Mobile Phone Review*' dataset to analyze the review as a positive review (1) or a negative review (0). The evaluation of the model was done using **5-fold cross-validation**.

## Dataset:

The dataset was a Mobile Phone Review dataset having reviews and their sentiments (1 for positive and 0 for negative). It consisted of **1000** reviews. The dataset was further split into 5 parts for the 5-fold cross-validation evaluation. In each step, $\frac{4}{5}^{th}$ of the dataset was used for training and $\frac{1}{5}^{th}$ was used for testing. The dataset is preprocessed by **removing punctuations** and **converting to lower case**.
We used the dataset in **3 formats**:

1. **Raw dataset**
2. **After removing stop words**
3. **After performing stemming**

## **Model**:

The Naive Bayes classifier uses the **naive assumption** that every word's occurrence is **independent** of the other. In each fold of the cross-validation, the **posterior probability** is calculated by calculating the **prior probability** and the **likelihood function** using the training data, and the posterior probability is used to classify the test data into positive or negative sentiment. Due to the fact that every word does not occur in every comment, we apply **Laplace smoothing** with a smoothing parameter of **1.** The Naive Bayes model is mathematically described as follows:

$$X = (x_1, x_2, x_3, ....., x_n)$$

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

# Results:

1. **Raw Data**

   Running Naive Bayes on raw data gave an accuracy of **80.6% ± 2.4%** and F-score **0.802 ± 0.031** after 5-fold cross-validations. This accuracy is good but was improved by removing stop words.

```
Accuracy after fold  1 :  0.83
Accuracy after fold  2 :  0.84
Accuracy after fold  3 :  0.78
Accuracy after fold  4 :  0.785
Accuracy after fold  5 :  0.795
Accuracy after 5-fold validation:  0.806  ±  0.024
F-score after 5-fold validation:  0.802  ±  0.031
```

2. **After removing stop words**

   Running Naive Bayes on the dataset after removing stop words gave an accuracy of **81.9% ± 2.6%** and F-score **0.814 ± 0.035** after 5-fold cross-validations. This accuracy is the **best** that we could get after removing unnecessary words and stop words.

```
Accuracy after fold  1 :  0.84
Accuracy after fold  2 :  0.855
Accuracy after fold  3 :  0.82
Accuracy after fold  4 :  0.79
Accuracy after fold  5 :  0.79
Accuracy after 5-fold validation:  0.819  ±  0.026
F-score after 5-fold validation:  0.814  ±  0.035
```

3. **Removed stop words + Stemming**

Running Naive Bayes on the dataset after removing stop words and stemming the words gave an accuracy of **81.1% ± 2.9%** and F-score **0.804 ± 0.039** after 5-fold cross-validations. This accuracy is better than the accuracy we got on the raw data set but not better than the accuracy we got after only removing stop words.

```
Accuracy after fold  1 :  0.83
Accuracy after fold  2 :  0.855
Accuracy after fold  3 :  0.805
Accuracy after fold  4 :  0.775
Accuracy after fold  5 :  0.79
Accuracy after 5-fold validation:  0.811  ±  0.029
F-score after 5-fold validation:  0.804  ±  0.039
```

## Conclusion:

We found that removing stop words but not performing stemming gave the best accuracy. This may be because stemming makes some words meaningless by removing '**s**' or **'ing'**. The complete result is summarized in the table below:

| | **Accuracy** | **F-Score** |
|---|---|---|
| **Raw Data** | 80.6% ± 2.4% | 0.802 ± 0.031 |
| **Removing Stop Words** | 81.9% ± 2.6% | 0.814 ± 0.035 |
| **Removing Stop Words + Stemming** | 81.1% ± 2.9% | 0.804 ± 0.039 |