# Naive Bayes Classifier

**Accuracy** = 0.9539527302363489 or 95.53%

**Auc Score** = 0.9346987864896824

**Features** = 34549

**Total document** = 12268

**Data Split** = 80% training / 20% testing

**Random State** = 3

**Confusion Matrix :**

|  | Negative | Positive |
| --- | --- | --- |
| Negative | 574 | 69 |
| Positive | 44 | 1767 |

## Logistic Regression Classifier

**Accuracy** = 0.9641401792991035 or 96.41%

**Auc Score** = 0.9911131078581913

**Features** = 28000

**Total document** = 12268

**Data Split** = 70% training / 30% testing

**Random State** = 3

**Solver** = ' liblinear '

**Confusion Matrix :**

|  | Negative | Positive |
| --- | --- | --- |
| Negative | 872 | 81 |
| Positive | 51 | 2677 |

## Module Used

* **Sklearn** : Scikit-learn is probably the most **useful library for machine learning in Python**. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
* **Pandas :** Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the [Python](https://www.python.org/) programming language.
* **Pickle :** Pickle is a module in Python used for serializing and de-serializing Python objects.
* **Nltk :** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](http://nltk.org/nltk_data/) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries
* **Re :** Regular expression in python

## Framework Used

Flask

## Important Notes

### **Feature extraction used** = CountVectorizer / Bag of words

### **Text processing used** :

* Lowercase
* Lemmatization
* Stopwords
* Tokenization
* Regular expression ( removing everything except letters )

### **Steps Involved :**

DataSet Collection

-> DataSet Modification

-> Text Preprocessing

-> Feature Extraction and Feature Selection

-> Model Training

-> Testing

-> Predict

-> Dump Model Pickle

-> Model Deployment Flask

### 

### **Measurement Techniques :**

* Accuracy score
* Confusion matrix
* Auc score

### **Bag Of Words / CountVectorizer Working :**

Consider the documents

- 'This is first document.','This is the second document.','And this is the third'

Two Steps involved :

* **Fit** : Learn a vocabulary dictionary of all tokens in the raw documents.

It returns the list of unique words present in the documents.

| **and** | **document** | **first** | **is** | **second** | **the** | **third** | **this** |
| --- | --- | --- | --- | --- | --- | --- | --- |

* **Transform :** Transform documents to document-term matrix.

It returns a sparse matrix with the word present in the document as 1 and not present as 0.

|  | **and** | **document** | **first** | **is** | **second** | **the** | **third** | **this** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **document1** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| **document2** | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **document3** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

### **Naive Bayes Derivation for Classfier :**

Formula :

| P ( A | B ) = P( B | A ) \* P( A ) / P( B ) |
| --- |

Dataset :

| X = { x1, x2, x3 …….xn } { y } |
| --- |
| | f1 | f2 | f3 | ….. | fn | y | | --- | --- | --- | --- | --- | --- | | x1 | x2 | x3 | …... | xn | y1 | |

Derivation :

| P ( y | x1, x2, x3…..xn ) = P( x1 | y )\* P( x1 | y )\* P( x1 | y ) ……..P( xn | y ) \* P(y)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    P(x1)\* P(x2)\* P(x3)......P(xn) -> Constant for all  P ( y | x1, x2, x3…..xn ) = P(y) ∑ P(xi | y)  P ( y | x1, x2, x3…..xn ) = argmax ( P(y) ∑ P(xi | y) ) -> main formula for classifier |
| --- |

### **Naive Bayes Classifier Working :**

Formula :

| P ( A | B ) = P( B | A ) \* P( A ) / P( B ) |
| --- |

Bag of words / CountVectoriser with output data :

| the | food | delicious | bad | output |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1(yes) |
| 1 | 1 | 0 | 1 | 0(no) |
| 0 | 1 | 0 | 1 | 0(no) |
|  | | | | |
| Sentence1 | The Food is delicious | | | |
| Sentence2 | The food is Bad | | | |
| Sentence3 | Food is Bad | | | |
|  | | | | |
| To Classify  Sentence - Good(yes) or Bad(no) | | | | |
| **For yes, p(y=yes) = 1/3, p(y=no) = 2/3**  **Sentence -** delicious food | | | | |
| p(y=yes) \* p(x1|y=yes) \* p(x2|y=yes)  ⅓ \* 1 \* ⅓ | | | | |
| **For no** | | | | |
| p(y=yes) \* p(x1|y=yes) \* p(x2|y=yes)  ⅔ \* 0 \* ⅔ | | | | |
| **The higher the value of probability of yes or no, the output would be selected from there.** | | | | |

### **Laplace Smoothing**