Naïve Bayes Classifier

Deadline March 16 2025

Problem Description

In this assignment, you will implement a Naïve Bayes classifier to classify text messages as spam or not spam. You will develop a data processing pipeline, train a probabilistic model, and evaluate its performance using various metrics.

The Naïve Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$ and $P(x|c)$. Naïve Bayes classifier assume class conditional independence which is the effect of the value of a preditor (x) on a given class (c) is independent of the values of other predictors. So the Posterior probability is:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{1}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times ... \times P(X_i|c) \times P(c) \tag{2}$$

where $P(c|x)$ is the posterior probability of class given attribute, $P(c)$ is the prior probability of class, $P(x|c)$ is the likelihood which is the probability of predictor given class, and $P(x)$ si the prior probability of predictor.

Modules Allowed

Only the following modules are allow to use in your code.

re
math
numpy
argparse
random
collections.defaultdict
collections.Counter
nltk.tokenize.word_tokenize
nltk.stem.PorterStemmer
nltk.corpus.stopwords

Instructions

0.1 Data Preprocessing (DataLoader Class)

Your class should include the following methods:

- preprocess():

  Convert text to lowercase. Remove special characters and punctuation. Tokenize words using word tokenize. Apply stemming using PorterStemmer. Remove stopwords.

- load_data():

  Read a text dataset (e.g., SMSSpamCollection). Extract labels and messages. Convert labels to binary format (spam = 1, ham = 0).

- split_data():

  Shuffle and split the dataset into training and testing sets (80%-20% split).

0.2 Implementing Naïve Bayes Classifier (NaiveBayes Class)

Your class should include the following methods:

- train():

  Compute class priors : $p(ham)$ and $p(spam)$. Count word occurrences separately for ham and spam messages. Compute probabilities of words given a class $p(word|ham)$, $p(word|spam)$ with Laplace smoothing.

- prediction():

  Compute the log probabilities of each message being ham or spam. Assign the label with the higher probability.

0.3 Evaluating Model Performance (EvaluationMetrics Class)

Your class should include the following methods:

- compute_metrics():

  Compute True Positive (TP), False Positives (FP), True Negatives (TN), False Negatives (FN). Compute Accuracy, Precision, Recall and F1 Score using the following equation:

  $Accuracy = \frac{TP+TN}{TP+TN+FN+FN}$

  $Precision = \frac{TP}{TP+FP}$

  $Recall = \frac{TP}{TP+FN}$

  $F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall}$

0.4 Logging and Analysis (main function

This step should be write in the $main()$ function.

Train the model using the training dataset. Predict both training and test samples. Compute and log performance metrics (results.log) including: Number of training and testing samples. Training and testing accuracy. Training and testing F1 Score. TP, TN, FP, FN values.

Deliverables

1. Python script implementing *DataLoader.py, NaiveBayes.py, EvaluationMetrics.py* and *Main.py*

2. Log file (results.log) containing model performance details.

3. Report (optional) discussing: Performance of the model. Challenges faced and solutions applied. Possible improvements.

Grading Criteria

Criteria	Points
Correct implementation of data preprocessing	20
Correct implementation of Naïve Bayes model	30
Correct computation of evaluation metrics	20
Logging and structured results	15
Code readability and documentation	15