| **PROJECT 2**  Harsha - idxxxx |
| --- |

| For this project, you are asked to implement a detection program supporting Short Message Service (SMS) spam filtering. The main concern is to design/generate features to differentiate SMS spam messages from legitimate ones, and run machine learning techniques (i.e., supervised learning) to classify SMS spam messages. Unlike email spam filtering, SMS spam filtering poses its own intrinsic problem because the length of text messages is relatively small (up to 160 characters or less). To come up with this project successfully, you must devise robust and efficient detection features to solve this problem.  **1. Data Reading and Preprocessing**  The SMS Spam collection contains a total of 1324 SMS messages, which is composed of 82 spam and 1002 legitimate messages. For the data preprocessing, feature extraction and .arff file creation, python programming language is used and the google colaboratory IDE was used. There are several libraries used for performing the above activities.  Below are the list of libraries installed :    Data Reading code is shown below :    Displaying the data :    Labeling the data based on whether it is a spam message or ham message :    **2. Feature Extraction**  There are four different features which are discussed in the project description which have to be implemented. The character count feature is implemented by applying length method. Currency symbol count is figured out by using regular expressions to find all the currency symbols. Numeric strings are also calculated by using regular expressions. Most popular word count was calculated using the lambda function by splitting the data using the most common function. Below shows the code below.    The column “Text” has been dropped and remaining features are the final list of features for the model building.  The final data frame will look like this :    Distribution for the features are shown below :    **2. Model Building in WEKA**  The data has been converted to .arff file in WEKA by uploading the .csv file into the ARFFViewer. This SMS\_Feature.arff is then loaded into the WEKA software for initial preprocessing. The same is shown below :      The above screenshot shows the distribution of Label features in the WEKA software where we have loaded the .arff file. It can be seen that the Ham records are 979 and Sam records are 277. Once the data is loaded, classifier models can be built from the next tab.  Below are the algorithms which are implemented in WEKA   1. **NaiveBayes Multinomial**       For Naive Bayes Multinomial algorithm, we have trained the model using the four features and the label as the target feature. We can see the evaluation metrics as below :   * Accuracy : 92.59% * True Positive Rate : 0.76 * False Positive Rate : 0.028   The Margin Curve also shows the training process.   1. **LibSVM**       For LibSVM, we have trained the model using the four features and the label as the target feature. We can see the evaluation metrics as below :   * Accuracy : 92.83% * True Positive Rate : 0.78 * False Positive Rate : 0.030  1. **Decision Trees (J48)**         For the Decision Tree (J48) algorithm, we have trained the model using the four features and the label as the target feature. We can see the evaluation metrics as below :   * Accuracy : 93.71% * True Positive Rate : 0.845 * False Positive Rate : 0.037   For this decision tree, the first node is taken as Numeric\_String\_Count which is then further branched down to currency symbol and most popular count based on the entropy value.   1. **Random Forest**       For the Random Forest algorithm, we have trained the model using the four features and the label as the target feature. We can see the evaluation metrics as below :   * Accuracy : 93.31% * True Positive Rate : 0.838 * False Positive Rate : 0.040 |
| --- |
| 1. **K-Nearest Neighbours**       For the K- Nearest Neighbors algorithm, we have trained the model using the four features and the label as the target feature. We can see the evaluation metrics as below :   * Accuracy : 93.31% * True Positive Rate : 0.856 * False Positive Rate : 0.045 |