**Spam Email Detection Project**

**INTRODUCTION :**

In today’s globalized world, email is a primary source of communication. This communication can vary from personal, business, corporate to government. With the rapid increase in email usage, there has also been increase in the SPAM emails. SPAM emails, also known as junk email involves nearly identical messages sent to numerous recipients by email. Apart from being annoying, spam emails can also pose a security threat to computer system. It is estimated that spam cost businesses on the order of $100 billion in 2007. In this project, we use text mining to perform automatic spam filtering to use emails effectively. We try to identify patterns using Data-mining classification algorithms to enable us classify the emails as HAM [0] or SPAM [1]

**LEARNING DATA :**

The data used for this project was taken from the Spam Assassin public corpus website. It consists of two data sets: train and test. Each dataset contains a randomly selected collection of emails in plain text format, which have been labelled as HAM [0] or SPAM [1]. The training data is used to build a model for classifying emails into HAM [0] and SPAM [1]. The test data is used to check the accuracy of the model built with the training data. The training data set contains 2893 rows X 3 columns emails with zero ham and zero spam emails. The test data contains subject, message , label.

**DATA PREPROCESSING :**

The emails in the learning data are in plain text format. We need to convert the plain text into features that can represent the emails. Using these features we can then use a learning algorithm on the emails. A number of pre-processing steps are first performed. We convert the plain text files to files with one word per line. In this project, we look at emails just as a collection of words.The output files are named as ‘message.csv’

**STOP WORDS :**

There are some English words which appear very frequently in all documents and so have no worth in representing the documents. These are called STOP WORDS and there is no harm in deleting them. Example: the, a, for etc. There are also some domain specific (in this case email) stop words such as mon, tue, email, sender, from etc. So, we delete these words from all the files using a duplicate.drop. These words are put in a file ‘words.txt’. The shell script takes multiple files as an argument and then deletes all the stop words mentioned in the words.txt file.

**STEMMING :**

The next step to be performed is stemming. Stemming is used to find a root of a word and thus replacing all words to their stem which reduces the number of words to be considered for representing a document. Example: sings, singing, sing have sing as their stem. In the project, we use python implementation of Porter stemming algorithm which is slightly modified to meet our needs. The resultant files are named with an extension ‘words\_stemmed’

In this project, I aim at implementing four spam filtering techniques that are widely used in various forms and as is. They are:

• Naïve Bayes Method for spam filtering

• K-Nearest Neighbors method for spam filtering

• Support Vector Machines

• LogisticRegression

• DecisionTreeClassifier

I shall be describing each of these methods in depth along with their implementations. I will also explain the methods used to implement them, the dataset used, the results in terms of accuracy and time taken as well as the drawbacks of each of them. Towards the end, I shall be discussing about the way any of these techniques have an upper hand over the others, if at all. Therefore, evaluating the techniques with respect to each other.

**NAÏVE BAYES ALGORITHM :**

The Bayesian Classification model, proposed by Thomas Bayes (1702 - 1761), is based on statistical and probabilistic method of learning. It is a type of supervised learning algorithm. Supervised learning is defined as the task of inferring a function from supervised training data. A supervised training algorithm analyzes the training data and comes up with the classifier function and the regression function. We are going to deal with classification here. The inferred classification function performs the task of predicting the correct output value for a given input value. The Bayesian classifier assumes a probabilistic underlying model and determines probabilities of the outcomes. It can solve diagnostic as well as predictive problems. Naïve Bayes is based on conditional probabilities. Conditional probability is the probability of the occurrence of an event, given that another event has occurred. For implementing Bayesian Classification for e-mail spam filtering, I used the Bayes theorem. The Bayes theorem in terms of spam and ham e-mails can be expressed as:

Pr(𝑆|𝑊) = Pr(𝑊|𝑆) Pr(𝑆) / Pr(𝑊|𝑠) Pr(𝑆) + Pr(𝑊|𝐻) Pr(𝐻)

Where Pr(S|W) is the probability that a message is spam knowing the presence of a given word in it. Pr(W|S) is the probability that a certain word appear in spam messages Pr(S) is the overall probability of a message being a spam message. Pr(W|H) is the probability that a certain word appears in ham messages. Pr(H) is the overall probability that a message is a ham message. As described in the implementation section, we have continuous attributes of categorical data. For classifying data with such attributes, I have used Gaussian Bayesian Filters. It is assumed that the continuous values are distributed along the Gaussian curve. The graph of a Gaussian is a symmetric bell shaped curve. To deal with such continuous attributes, let us say, training data x, We first segment the data by class. Then we have to compute the mean and variance of each class.

**K- NEAREST NEIGHBORS :**

The k Nearest Neighbors algorithm, also known as kNN, is a non-parametric method used for pattern classification. It is an instance-based learning algorithm. The function is simply approximated locally. All computation is deferred until classification. The neighbors for the data points are taken as a set of objects whose values are known. This is basically the training phase of the algorithm. In the classification phase of the algorithm, for each point in the testing set, we consider k neighbors where k is predefined by the programmer. We then find the most frequent class values in the training set. We can find the neighbors using many methods such as linear search, space partitioning, locality sensitive hashing etc. For implementing the algorithm, I have used the Euclidean distance between the attributes of the instances. The squared Euclidean distance between the point p = (p1, p2, ..., pn) and the point q = (q1, q2, ..., qn) is the sum of the squares of the differences between the components: Dist2 (p,q) = Σi (pi – qi) .

The Euclidean distance is then the square root of Dist2 (p,q)[. While Euclidean distance is a good measure for finding all the neighbors of a given instance, it has a major drawback that it cannot work for very large data sets as the computational time is very high. I am going to demonstrate this by implementing it on the 4091 instances that I have in the dataset. Figure 3: Plot depicting 5 nearest neighbors of x The above figure depicts the general idea of the k Nearest Neighbors algorithms. There are two classes, blue and black. We have a data point x that needs to be classified. Considering that the value of k is taken as 5, we first find the distance of x from all of the given points. All the distances are then sorted in ascending order and the first 5 distances are considered. We are thus considering the 5 points that are nearest to point x. We then check the maximum frequency of points within the neighborhood of x. As we can see, there are 3 black points and 2 blue points. So the maximum frequency is 3 and it belongs to black. X is therefore classified as a black point. To implement the kNN-algorithm, my program divides the spam base into a testing dataset and a training dataset. There is no preprocessing on the training dataset. We directly calculate the k nearest neighbors for each data point in the testing dataset. Then among these k values, I calculate whether the email is mostly classified as ham or spam. If majority of the k neighbors are classified as spam, the data point or the instance is classified as spam. Otherwise it is classified as ham. Then we move on to calculate the accuracy by checking the value of the ham/spam to the original class value.

**SUPPORT VECTOR MACHINES :**

Support Vector Machines, also called as SVM are supervised learning models that analyze data and recognize patterns. If we have a set of training data points, each marked with a class value such as [0,1] , [1,-1] , [TRUE,FALSE] , on being fed with new data points, a support vector machine can categorize them into one of the two classes. Therefore, a support vector machine is a non-probabilistic binary linear classifier. How a support vector machine works is, it produces a hyper plane on a high dimensional space or a set of hyper planes on an infinite dimension space. Classification of data points is done around these hyper planes. For two classes with linearly separable data, there may be many such separators that can separate the two classes of data. Intuitively, it seems like a decision boundary that is drawn in the middle of the void between the two classes seems to be the best positioning of the decision boundary.

Consider the figure above, the line H1 is a bad choice for a decision boundary because it does not separate the black points and the white points into two different classes. Therefore, H1 is eliminated as an option. Now consider the line H2. H2 is a valid decision boundary, but not a good one. The reason being, the margin While some learning methods such as the perceptron algorithm find just any linear separator, others, like Naive Bayes, search for the best linear separator according to some criterion. The SVM defines the criterion to be looking for a decision surface that is maximally far away from data points in either of the two linearly separable classes. The margin is this distance which is the distance between the nearest point to either of the classes and the hyper plane that linearly separates them. This method of construction necessarily means that the decision function for an SVM is fully specified by a (usually small) subset of the data which defines the position of the separator. The points, that lie on the hyper plane are called as support vectors. Other data points play no part in determining the decision surface that is chosen. Margin maximization in support vector machines The figure given below explains and depicts the term support vectors and margin.

**LOGISTICREGRESSION :**

The logistic model  is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

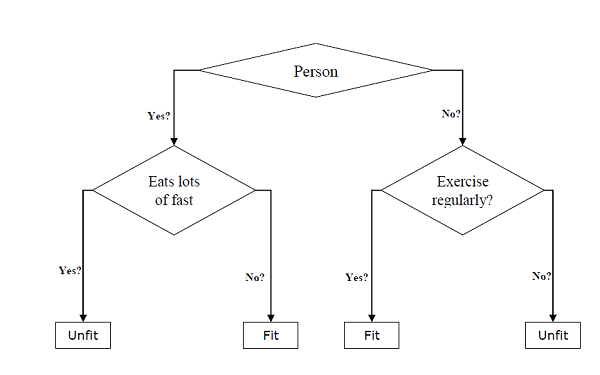
Logistic regression is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that in its basic form uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), although many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), **logistic regression**[[1]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-1) (or **logit regression**) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (a form of [binary regression](https://en.wikipedia.org/wiki/Binary_regression)). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labeled "0" and "1". In the logistic model, the [log-odds](https://en.wikipedia.org/wiki/Log-odds) (the [logarithm](https://en.wikipedia.org/wiki/Logarithm) of the [odds](https://en.wikipedia.org/wiki/Odds)) for the value labeled "1" is a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of one or more [independent variables](https://en.wikipedia.org/wiki/Independent_variable) ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a [continuous variable](https://en.wikipedia.org/wiki/Continuous_variable) (any real value). The corresponding [probability](https://en.wikipedia.org/wiki/Probability) of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The [unit of measurement](https://en.wikipedia.org/wiki/Unit_of_measurement) for the log-odds scale is called a [*logit*](https://en.wikipedia.org/wiki/Logit), from ***log****istic un****it***, hence the alternative names. Analogous models with a different [sigmoid function](https://en.wikipedia.org/wiki/Sigmoid_function) instead of the logistic function can also be used, such as the [probit model](https://en.wikipedia.org/wiki/Probit_model" \o "Probit model); the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a *constant* rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the [odds ratio](https://en.wikipedia.org/wiki/Odds_ratio).

In a binary logistic regression model, the dependent variable has two levels ([categorical](https://en.wikipedia.org/wiki/Categorical_variable)). Outputs with more than two values are modeled by [multinomial logistic regression](https://en.wikipedia.org/wiki/Multinomial_logistic_regression) and, if the multiple categories are [ordered](https://en.wikipedia.org/wiki/Level_of_measurement#Ordinal_type), by [ordinal logistic regression](https://en.wikipedia.org/wiki/Ordinal_logistic_regression) (for example the proportional odds ordinal logistic model[[2]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-wal67est-2)). The logistic regression model itself simply models probability of output in terms of input and does not perform [statistical classification](https://en.wikipedia.org/wiki/Statistical_classification) (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a [binary classifier](https://en.wikipedia.org/wiki/Binary_classifier). The coefficients are generally not computed by a closed-form expression, unlike [linear least squares](https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics)), see [and Model fitting](https://en.wikipedia.org/wiki/Logistic_regression#Model_fitting).

**DecisionTreeClassifier :**

Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions tress are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below −



In the above decision tree, the question are decision nodes and final outcomes are leaves. We have the following two types of decision trees −

**Classification decision trees** − In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.

**Regression decision trees** − In this kind of decision trees, the decision variable is continuous.

Implementing Decision Tree Algorithm

Gini Index

It is the name of the cost function that is used to evaluate the binary splits in the dataset and works with the categorial target variable “Success” or “Failure”.

Higher the value of Gini index, higher the homogeneity. A perfect Gini index value is 0 and worst is 0.5 (for 2 class problem). Gini index for a split can be calculated with the help of following steps −

First, calculate Gini index for sub-nodes by using the formula p^2+q^2 , which is the sum of the square of probability for success and failure.

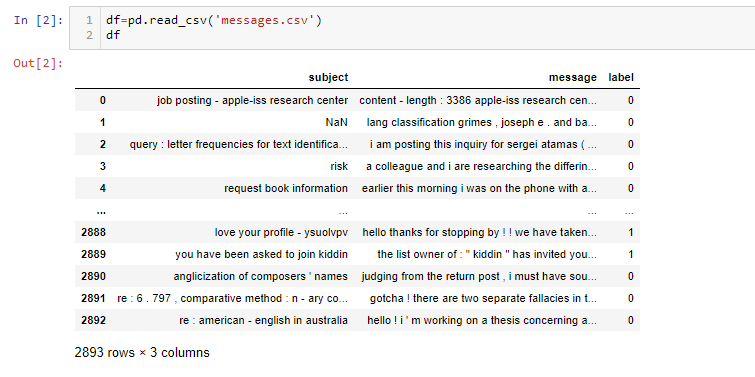
Next, calculate Gini index for split using weighted Gini score of each node of that split.

Classification and Regression Tree (CART) algorithm uses Gini method to generate binary splits.

**IMPLEMENTATION IN PYTHON :**

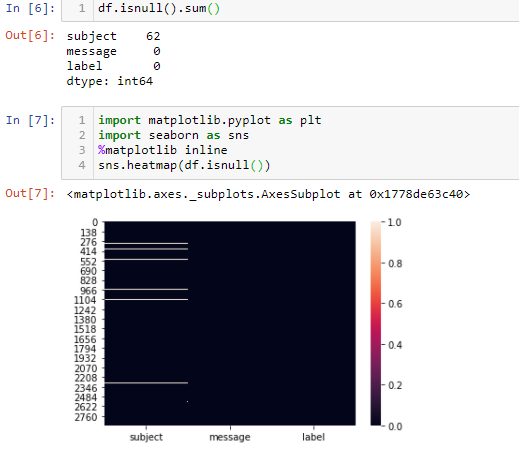


Next, download the iris dataset as follows –

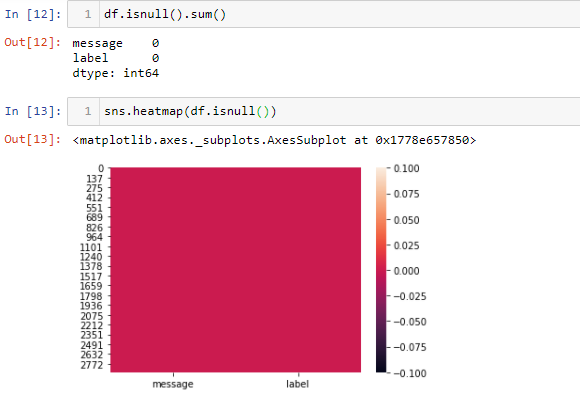


Now, checking the shape , columns and null value in dataset−





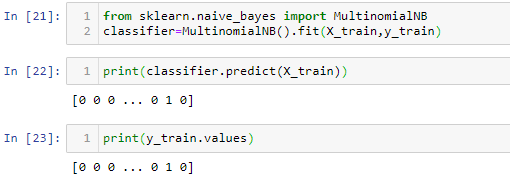
Now , cleaning the data and droping the subject column and we getting all null values –



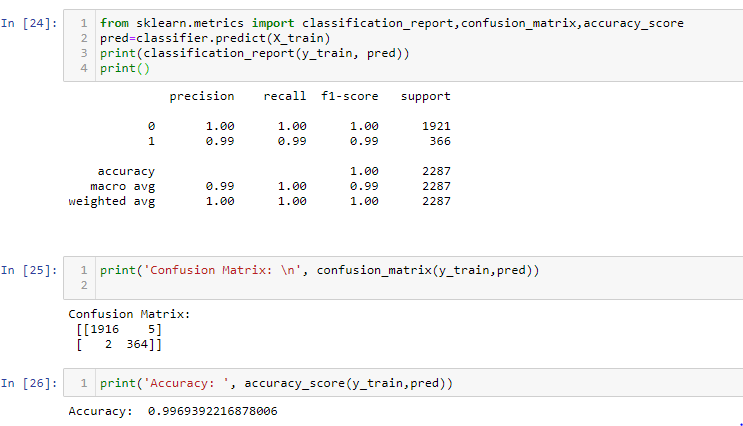
Now, split the dataset into features and target variable as follows –



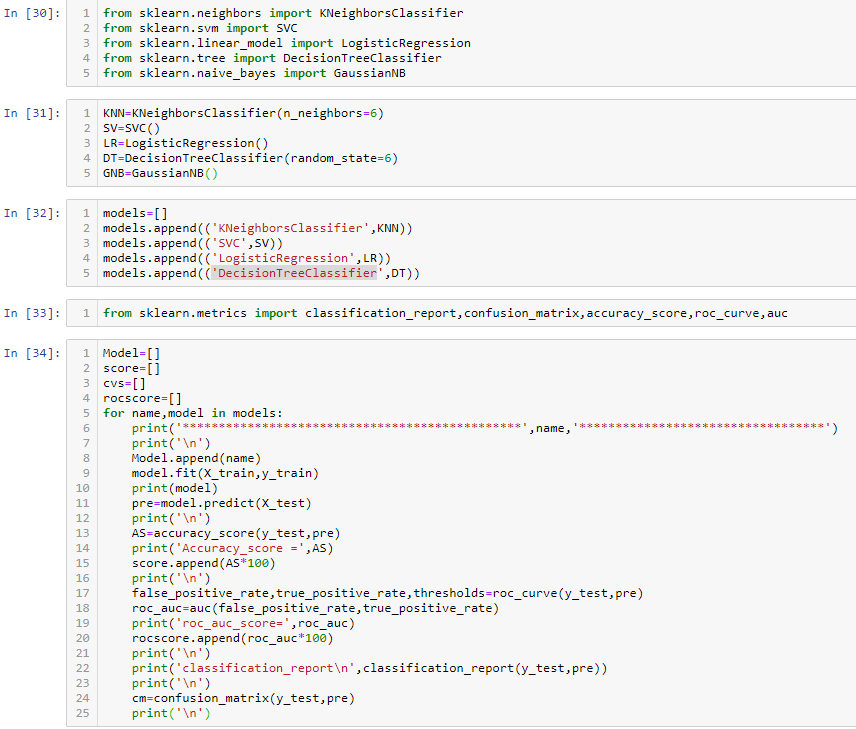
Next, train the model with the help of MultinomialNB class as follows –



At last we need to make prediction. It can be done with the help of following script –



Next, we can get the accuracy score, confusion matrix and classification report as follows −



**OUTPUT :**

