**PROJECT REPORT**

(Project Term August-December 2021)

## (TITLE OF THE PROJECT)

Submitted by

**ASHUTOSH KUMAR Registration Number: 11916550**

**Course Code INT246**

Under the Guidance of

# Dr.Sagar Pande

# School of Computer Science and Engineering

A picture containing text

Description automatically generated

**DECLARATION**

We hereby declare that the project work entitled **“Spam Email Classification”** is an authentic record of our own work carried out as requirements of Project for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Dr. Sagar Pande, during August to November 2020. All the information furnished in this project report is based on our own intensive work and is genuine.

Name of Student: Ashutosh Kumar

Registration Number:11916550

Ashutosh Kumar

Date:10/12/21

**CERTIFICATE**

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfilment of the conditions for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

**Signature and Name of the Mentor**

**Designation**

**School of Computer Science and Engineering,**

Lovely Professional University,

Phagwara, Punjab.

Date :

|  |  |  |
| --- | --- | --- |
| S.No | Topic | Page no |
| 1. | Introduction | 5 |
| 2. | Learning Data | 6 |
| 3. | Stop Words | 6 |
| 4. | Process Flow | 6 |
| 5. | Working | 6 |
| 6. | Algorithms | 6-22 |
| 7. | Testing | 23 |
| 8. | Performance Measure | 23 |
| 9. | Program | 24-29 |
| 10. | Changes Done | 30 |
| 11. | Future Scope | 30 |
| 12. | Conclusion | 30 |
| 13. | Reference | 31 |

# 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. The word ‘Spam’ as applied to Email means “Unsolicited Bulk Email”. Unsolicited means that the recipient has not granted verifiable permission for the message to be sent. Bulk means that the message is sent as a part of a larger collection of messages, all having substantively identical content. Spam is an issue about consent, not content. Whether the Unsolicited Bulk Email message is an advert, a scam, a begging letter or an offer letter of something is provided to you is free of cost, the content is irrelevant- if the message was sent unsolicited and in bulk then the message is spam.

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 or SPAM.

# LEARNING DATA

The data used for this project was taken from the pre-refined Spam Assassin from Kaggle website. Each dataset contains a randomly selected collection of emails , which have been labelled as 0 for HAM or 1 for SPAM.

This is a csv file containing related information of 5172 randomly picked email files and their respective labels for spam or not-spam classification.

The csv file contains 5172 rows, each row for each email. There are 3002 columns. The first column indicates Email name. The name has been set with numbers and not recipients' name to protect privacy. The last column has the labels for prediction: 1 for spam, 0 for not spam. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding the non-alphabetical characters/words. For each row, the count of each word(column) in that email(row) is stored in the respective cells*.* Thus, information regarding all 5172 emails are stored in a compact dataframe rather than as separate text files.

# DATA PREPROCESSING

The dataset that I used is already pre-refined. With Label Encoder the words of the email has been given numerical values after that with tokenization it is converted into tokens and with the help of Porter Stemmer the words got stemmed (means words with ing, ed, ner at the end got converted into their root word or has been removed. Example winning gets converted into win). This helps to reduce the memory usage by the model. As these processes has already been done, ther eare some minor changes that need to be done. Like removing Stop Words.

## 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, I deleted these word columns from all the dataset and copied it to new dataframe.

# PROCESS FLOW

The whole process of classification is depicted in the following diagram:

1. Collecting Data
2. Data Pre-processing
3. Splitting the Dataset into training and testing data
4. Fitting the training data into numerous models
5. Performance Measuring
6. Parameter Tuning using GridSearchCV

# WORKING

The training data set are given as an input to different classifying algorithms such as Naïve Bayes, Bayes Net, Neural Network, k-NN and Decision Table were used to build models. Then, the test data set were tested to find the classification accuracy of each model. The k-NN method gives the highest classification accuracy using k=3, cross-validation with 10 folds and 20 attributes. The accuracy came up to 94.5 %.

**ALGORITHMS USED**

**NAÏVE BAYES**

## What is Naive Bayes algorithm?

It is a [classification technique](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/09/Bayes_rule-300x172.png)Above,

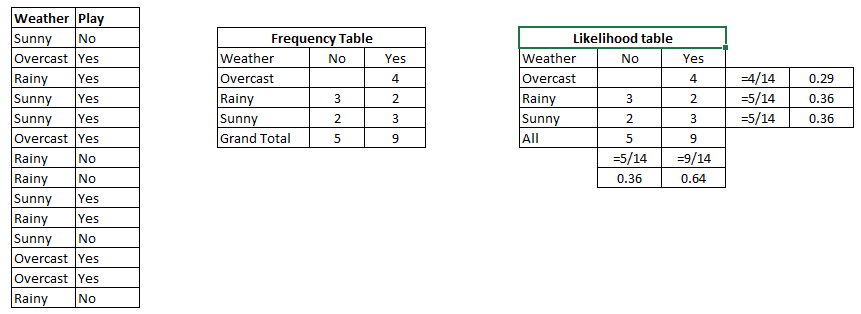
* *P*(*c|x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
* *P*(*c*) is the prior probability of *class*.
* *P*(*x|c*) is the likelihood which is the probability of *predictor* given *class*.
* *P*(*x*) is the prior probability of *predictor*.

## How Naive Bayes algorithm works?

Let’s understand it using an example. Below I have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/08/Bayes_41.png)

Step 3: Now, use [Naive Bayesian](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained) equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

**Problem:**Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P( Sunny | Yes) \* P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P( Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 \* 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

***Pros:***

* It is easy and fast to predict class of test data set. It also perform well in multi class prediction
* When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
* It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

***Cons:***

* If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
* On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.
* Another limitation of [Naive Bayes](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained) is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

## 4 Applications of Naive Bayes Algorithms

* **Real time Prediction:**Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
* **Multi class Prediction:**This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
* **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
* **Recommendation System:**Naive Bayes Classifier and [Collaborative Filtering](https://en.wikipedia.org/wiki/Collaborative_filtering) together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

Again, scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under the scikit-learn library:

* [**Gaussian:**](http://scikit-learn.org/stable/modules/naive_bayes.html)It is used in classification and it assumes that features follow a normal distribution.
* [**Multinomial**](http://scikit-learn.org/stable/modules/naive_bayes.html)**:**It is used for discrete counts. For example, let’s say,  we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, you can think of it as “number of times outcome number x\_i is observed over the n trials”.
* [**Bernoulli**](http://scikit-learn.org/stable/modules/naive_bayes.html)**:**The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

**XGBOOST**

XGBoost is an implementation of Gradient Boosted decision trees. This library was written in C++. It is a type of Software library that was designed basically to improve speed and model performance. It has recently been dominating in applied machine learning. XGBoost models majorly dominate in many Kaggle Competitions.

Chart

Description automatically generated

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. Weight of variables predicted wrong by the tree is increased and these the variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

**XGBoost Features**  
The library is laser-focused on computational speed and model performance, as such, there are few frills.  
**Model Features**  
Three main forms of gradient boosting are supported:

* Gradient Boosting
* Stochastic Gradient Boosting
* Regularized Gradient Boosting

**DECISION TREE**

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

**Diagram

Description automatically generated**

A tree can be *“learned”* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called*recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf(in this case YES/NO). *(Outlook = Rain, Temperature = Hot, Humidity = High, Wind = Strong )* would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance.

In other words we can say that decision tree represent a disjunction of conjunctions of constraints on the attribute values of instances.

*(Outlook = Sunny ^ Humidity = Normal) v (Outlook = Overcast) v (Outlook = Rain ^ Wind = Weak)*

**Pros**

* Decision trees are able to generate understandable rules.
* Decision trees perform classification without requiring much computation.
* Decision trees are able to handle both continuous and categorical variables.
* Decision trees provide a clear indication of which fields are most important for prediction or classification.

Cons: 

* Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
* Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
* Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

**RANDOM FOREST**

A random forest is a machine learning technique that’s used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The ‘forest’ generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like [scikit-learn](https://en.wikipedia.org/wiki/Scikit-learn)).

### **Features of a Random Forest Algorithm**

* It’s more accurate than the decision tree algorithm.
* It provides an effective way of handling missing data.
* It can produce a reasonable prediction without hyper-parameter tuning.
* It solves the issue of overfitting in decision trees.
* In every random forest tree, a subset of features is selected randomly at the node’s splitting point.

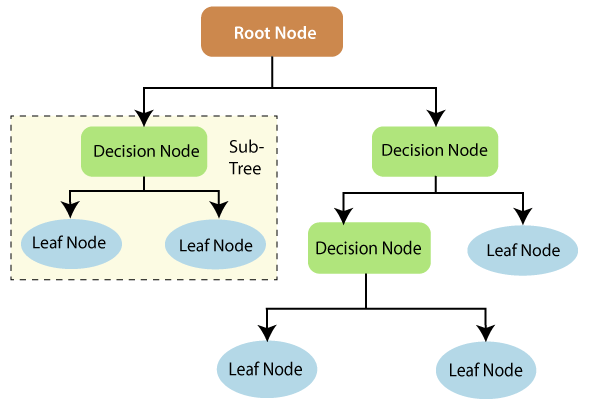
### **How random forest algorithm works**

#### **Understanding decision trees**

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.



[Image Source: Tutorials and Example](https://www.tutorialandexample.com/wp-content/uploads/2019/10/Decision-Trees-Root-Node.png)

The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees. An overview of these fundamental concepts will improve our understanding of how decision trees are built.

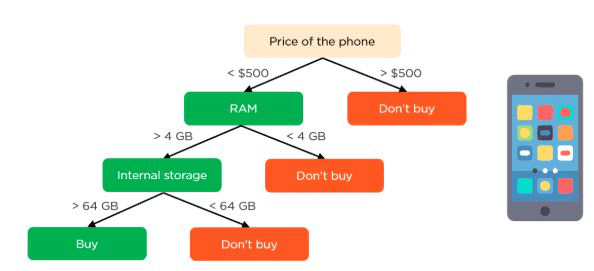
Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the [conditional entropy](https://en.wikipedia.org/wiki/Conditional_entropy) of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Let’s take a simple example of how a decision tree works. Suppose we want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.

The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either buying or not buying. The main features that determine the choice include the price, internal storage, and Random Access Memory (RAM). The decision tree will appear as follows.



[Image Source: Simplilearn](https://www.simplilearn.com/ice9/free_resources_article_thumb/phone-price.JPG)

#### **Applying decision trees in random forest**

The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction.

Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

Our first example can still be used to explain how random forests work. Instead of having a single decision tree, the random forest will have many decision trees. Let’s assume we have only four decision trees. In this case, the training data comprising the phone’s observations and features will be divided into four root nodes.

The root nodes could represent four features that could influence the customer’s choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees.

The outcome chosen by most decision trees will be the final choice. If three trees predict buying, and one tree predicts not buying, then the final prediction will be buying. In this case, it’s predicted that the customer will buy the phone.

**SVM**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

## Types of SVM

**SVM can be of two types:**

* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**K NEARES NEIGHBORS**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

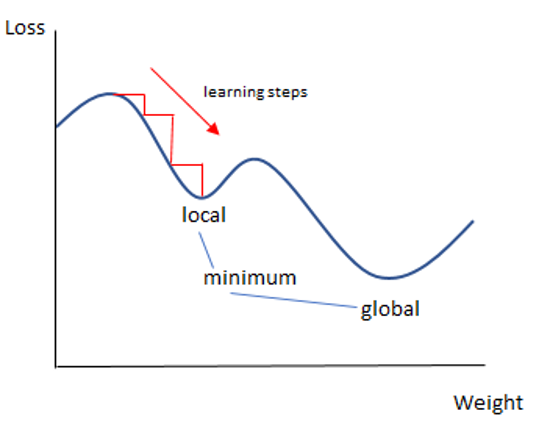
**SGD CLASSIFIER**

The name Stochastic Gradient Descent - Classifier (SGD-Classifier) might mislead some user to think that SGD is a classifier. But that’s not the case! SGD Classifier is a linear classifier (SVM, logistic regression, a.o.) optimized by the SGD. These are two different concepts. While SGD is a optimization method, Logistic Regression or linear Support Vector Machine is a machine learning algorithm/model. You can think of that a machine learning model defines a loss function, and the optimization method minimizes/maximizes it.

# 2 Background information on SGD Classifiers

**Gradient Descent**

First of all let’s talk about Gradient descent in general.



In a nutshell gradient descent is used to minimize a cost function. Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. But we can also use these kinds of algorithms to optimize our linear classifier such as Logistic Regression and linear Support Vecotor Machines.

There are three well known types of gradient decent:

1. Batch gradient descent
2. Stochastic gradient descent
3. Mini-batch gradient descent

Batch gradient descent computes the gradient using the whole dataset to find the minimum located in it’s basin of attraction.

Stochastic gradient descent (SGD) computes the gradient using a single sample.

Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples.

# TESTING STATISTICS

The following data shows the testing statistics of different models on the test data. The data shown here is for the frequency method used for attribute value representation with 20 attributes and cross validation with 10 folds.

## Naïve Bayes

Classification Accuracy: 81%

== Confusion Matrix ===

a b <-- classified as 56 5 | a = spam

33 106 | b = ham

## Bayesian Network

Classification Accuracy: 80%

=== Confusion Matrix ===

a b <-- classified as 58 3 | a = spam

37 102 | b = ham

## Neural Network

Classification Accuracy: 93.5 %

=== Confusion Matrix ===

a b <-- classified as

56 5 | a = spam

8 131 | b = ham

## SMO

Classification Accuracy: 88.5 %

=== Confusion Matrix ===

a b <-- classified as

47 14 | a = spam

9 130 | b = ham

## k-NN (k=1)

Classification Accuracy: 91.25 %

=== Confusion Matrix ===

a b <-- classified as 102 15 | a = spam

20 263 | b = ham

## k-NN (k=3)

Classification Accuracy: 94.5 %

=== Confusion Matrix ===

a b <-- classified as 58 3 | a = spam

8 131 | b = ham

## Decision Table

Classification Accuracy: 88 % === Confusion Matrix === a b <-- classified as 45 16 | a = spam

8 131 | b = ham

## C 4.5

Classification Accuracy: 90.5 %

=== Confusion Matrix ===

a b <-- classified as 57 4 | a = spam

15 124 | b = ham

# EXPERIMENTAL RESULTS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No:** | **Models** | **Accuracy Score** | **Computation Time (in seconds)** | **Accuracy Score by Cross Validation Method** | **Computation Time for cross validation** |
| 1. | SVC | 80.4 | 25.1 | X | X |
| 2. | Linear SVC | 95.10 | 0.45 | X | X |
| 3. | Multi-Layer Perceptron | 94.7 | 26.5 | 97.97 | 122.67 |
| 4. | Multinomial Naïve Bayes | 94.72 | 0.14 | 96.4 | 0.78 |
| 5. | Gaussian Naïve Bayes | 94.96 | 0.42 | 96.9 | 2.05 |
| 6. | K Neighbors Classifier | 85.29 | 1.10 | X | X |
| 7. | Decision Tree Classifier | 92.8 | 0.73 | 92.35 | 5.01 |
| 8. | SGD Classifier | 93.55 | 0.67 | 95.55 | 3.55 |
| 9. | Random Forest Classifier | 97.24 | 2.11 | 97.87 | 13.60 |
| 10. | Perceptron | 94.9 | 0.71 | 94.5 | 3.15 |
| 11. | XGBClassifier | 97.1 | 9.31 | 97.5 | 47.62 |
| x | x | x | x | x | x |

# PROGRAMS

A screenshot of a computer

Description automatically generated

**Graphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedText, application, email

Description automatically generatedText

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**CHANGES DONE AFTER VIVA**

* Removed Regression Models
* Replaced large chunk of code with simpler one
* Added computation time
* Used confusion matrix as suggested
* Added Perceptron, K Neighbors Classifier, SGDClassifier, LinearSVC

**FUTURE SCOPE**

* Use Bigger dataset or combine different dataset
* Can be used for spam message classification
* Use Feature Engineering
* Anomaly Detection
* Lemmatization
* Use Deep learning algorithms

# CONCLUSION

Given a set of words, we used feature selection to obtain words which allo us to distinguish between spam and ham emails. We also compared the accuracy of various classifiers in predicting the class attribute. We see that Multi-Layer Perceptron method gives the highest classification accuracy97.97% no matter how many attributes are used and which method is used. Also, Random Forest classifier gives 97.87% and XGBClassifier gives 97.1% accuracy score. We also see that on average the accuracy improves as the number of attributes increase. It is possible that the accuracy may increase more than 98% if we make further changes in parameters of the following models.

**REFERENCES**

* Porter, M (n.d). *The Porter Stemming Algorithm*. Retrieved March 2011, from http://tartarus.org/~martin/PorterStemmer

* Stack Overflow for removing errors
* Code for Cause YouTube Channel (Machine Learning Bootcamp)
* BasicCode YouTube Channel (Machine Learning Tutorials)
* Analytics Vidhya Blogs for various Models
* Data Science made Easy Blogs
* Sklearn Documentation
* TensorFlow Documentation
* Geeks for Geeks