Articles Journal using Data Mining

Prepared By:

1. **Youssof Kamal Eldin**
2. **Abdelrahman Salah Abdelrazzak**
3. **Abdelhady Eslam Mohamed**
4. **Mahmoud Hamed Hafez**
5. **Abdelrahman Nasser Ahmed**
6. **Mohamed sayed Saad**

Under supervision:

Prof. Marghny Hassan Mohamed

Contents

List of Figures

Abstract

Data mining can be defined as the process of finding previously unknown patterns , trends in databases and using that information to build predictive models.

Alternatively , it can be defined as the process of data selection and exploration and building models using vast data stores to uncover previously unknown pattern. Data mining in not new it has been used intensively and extensively by financial institution, for credit scoring and fraud detection. Another factor is that the huge amounts of data generated by Article sentiment analysis transaction are too complex and voluminous to be processed and analyzed by traditional methods.

This project aims for mining the relationship of some Article for classification (positive/negative).The data mining methods and techniques will be explored to identify the suitable methods and techniques for efficient classification of Article and in mining useful patterns.

Our goals

Save time and effort
Find the filed of (positive / negative) for article
Making the decision-making in some things easier

Chapter 1

Introduction

1.1 Overview

our website is called "Articles Blog" it is a website for writing and analyzing articles about Economy and finding positive / negative classified articles based on Natural Language Processing in this website user can:-

– see all articles in the website.
– search for articles by:

– Tags.
– Opinions "This article is positive or negative".
– Words "That if the user wants to search for articles that have specific words".

– make new article with its title and can add tags for the article.
– Like / Dislike article and Comment on it.

1.2 Project Motivation:

Users need positive / negative opinions about specific subject.
Users spent time and effort for reading all available articles.

The difficulty of the process of the articles revisions because of its huge number.

1.3 Project Objective:

from these points we have mentioned in the motivation we realized that we are really need a solution to help us to overcome these problems. to overcome these problems we will go to make a website to predict the status of the article (positive / negative) using some of the methods of data mining algorithms.

1.4 Data Sets Used

1.4.1 Economic News Article Tone and Relevance

Contributors read snippets of news articles. They then noted if the article was relevant to the US economy and, if so, what the tone of the article was. Tone was judged on a 9 point scale (from 0 to 9, with 0 representing absolutely negative). Dataset contains these judgments as well as the dates, source titles, and text. Dates range from 1951 to 2014.

Number Of Articles : 8000 Articles

Number of Attributes : 15 Attributes including Positivity field

Attributes Information :

– Unit id:- It has id for each article
– Golden :- FALSE
– Unit state :- Finalized

– Trusted judgments :- 3
– Last judgment at :- DATE
– Positivity :- From 0 to 9
– Positivity : Confidence :- 0 to 1
– Relevance :- Yes , No , not sure
– Relevance : Confidence :- 0 to 1
– Article id :- Serial number
– Date :- DATE
– Headline :- Title
– Text : main article

Class :

– Positive
– Negative

1.4.2 U.S. economic performance based on news articles

Contributors viewed a new article headline and a short, bolded excerpt of a sentence or two from the attendant article. Next, they decided if the sentence in question provided an indication of the U.S. Economy health, then rated the indication on a scale of 0-9, with 0 being negative and 9 being positive.

Number Of Article : 5000 Article

Number of Attributes : 17 Attributes including Positivity field

Attributes Information :

– Unit id:- It has id for each article
– Golden :- FALSE
– Unit state :- Finalized
– Trusted judgments :- 3

– Last judgment at :- DATE
– Positivity :- From 0 to 9
– Positivity : Confidence :- 0 to 1
– Relevance :- Yes , No , not sure
– Relevance : Confidence :- 0 to 1
– Article id :- Serial number
– Date :- DATE
– Headline :- Title
– Line id
– Next sentence
– Previous sentence
– Text

Class :

– Positive
– Negative

1.5 Tools

1.5.1 Language

Python

1.5.2 DataBases

MYSQL

MySQL is a very popular, open source DBMS MySQL databases are relational

MONGODB

MONGODB is a very popular, open source RDBMS MONGODB databases are NON relational

1.5.3 BackEnd (Django PlatForm)

– it is python based platform for web development
– Loose coupling
– Less code.
– Quick development.
– Don’t repeat yourself (DRY).
– Explicit is better than implicit.
– Consistency.

1.5.4 Front End

HTML :

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them, the browser does not display the HTML tags, but uses them to determine how to display the document.

CSS :

CSS stands for Cascading Style Sheets, CSS saves a lot of work It can control the layout of multiple web pages all at once.

HTML was never intended to contain tags for formatting a web page , HTML was created to describe the content of a web page, like:

```
<h1>This is a heading</h1>

<p>This is a paragraph.</p>
```

When tags like *font*, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

JAVA SCRIPT :

One of many JavaScript HTML methods is

```
getElementById()
```

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (innerHTML) to "Hello JavaScript"

```
getElementById('demo').innerHTML = "Hello
JavaScript";
```

JQUERY :

– jQuery is a lightweight, "write less, do more", JavaScript library.
– The purpose of jQuery is to make it much easier to use JavaScript on your website.
– jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
– jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
– The jQuery library contains the following features:

– HTML/DOM manipulation
– CSS manipulation
– HTML event methods
– Effects and animations
– AJAX
– Utilities

BOOTSTRAP :

– Bootstrap is a free front-end framework for faster and easier web development.
– Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.
– Bootstrap also gives you the ability to easily create responsive designs.
– There are two ways to start using Bootstrap on your own web site
– Download Bootstrap from getbootstrap.com
– Include Bootstrap from a CDN
We use the first way

1.5.5 Classification Algorithms

the classification algorithms we used in this project are:
– Logistic Regression
– K Nearest Neighbors
– Support Vector Classifier
– Decision Tree
– Random Forests

chapter 2

Sentiment Analysis

Sentiment Analysis or Opinion Mining refers to the use of NLP, text analysis and computational linguistics to determine subjective information or the emotional state of the writer/subject/topic. It is commonly used in reviews which save businesses a lot of time from manually reading comments.

2.1 Data Preprocessing

You cannot go straight from raw text to fitting a machine learning or deep learning model
You must clean your text first, which means splitting it into words and handling punctuation and case.
In fact, there's a whole suit of text preparation methods that you mat need to use, and the choice of methods really depends on your NLP task.

This chapter has three sections, listed

- **Language**
- **library**
- **Steps**

2.1.1 Language :

We use **Python** ..why?
The language is easy and widespread and has many libraries.

2.1.2 library :

- **re**
- **pandas**
- **Stopwords from nltk.corpus**
- **word_tokenize from nltk.tokenize**
- **PorterStemmer from nltk.stem.porter**

2.1.3 Steps

– Split into Words

NLTK provides a function called word_tokenize() for splitting strings into tokens (nominally words). It splits tokens based on white space and punctuation. For example, commas and periods are taken as separate tokens. Contractions are split apart (e.g. “What’s” becomes “What” “‘s“). Quotes are kept, and so on.

```
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)
print(tokens[:100])
```

– Filter Out Punctuation

We can filter out all tokens that we are not interested in, such as all standalone punctuation.
This can be done by iterating over all tokens and only keeping those tokens that are all alphabetic. Python has the function isalpha() that can be used. For example:

```
# remove all tokens that are not alphabetic
words = [word for word in tokens if word.isalpha()]
print(words[:100])
```

– Filter out Stop Words (and Pipeline)

The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

– Stop Words:

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been

programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK(Natural Language Toolkit) in python has a list of stop words stored in 16 different languages. You can find them in the nltk_data directory.

Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
print(stop_words)
```

it print: ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', ...]

– Normalizing Case

– It is common to convert all words to one case.
– This means that the vocabulary will shrink in size, but some distinctions are lost (e.g. "Apple" the company vs "apple" the fruit is a commonly used example).
– We can convert all words to lowercase by calling the lower() function on each word.
– For example:

```
# convert to lower case
tokens = [w.lower() for w in tokens]
```

– Stem Words:

– Stemming refers to the process of reducing each word to its root or base.
– For example "fishing", "fished", and "fisher" all reduce to the stem "fish".
– Some applications, like document classification, may benefit from stemming in order to both reduce the vocabulary and to focus on the sense or sentiment of a document rather than deeper meaning.
– There are many stemming algorithms, although a popular and long-standing method is the Porter Stemming algorithm.
– This method is available in NLTK via the PorterStemmer class.
– For example:

```
# stemming of words
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()
stemmed = [porter.stem(word) for word in tokens]
print(stemmed[:100])
```

– Count Vectorize:

Dataset 'text' column can be viewed as a a big chunk of text separated by spaces and feed-back and carriage-returns '\r', '\n' and ' ', anything else is words and punctuation. when vectorizing text we simply create a big vector, at least big enough, with its size equal to different words in text. We here are considering words after all steps above are applied(splitting, removing punctuation, normalizing, and stemming). With this vector, every text is expressed as a vector with its length same as the big vector. where each element represent count of a word in this specific text.

```
from sklearn.feature_extraction.text
import CountVectorizer
v = CountVectorizer(analyzer = "word")
X_train_features =
v.fit_transform(X_train)
```

2.1.4 Summary

All what has been discussed can above can be summarized with the full Data Preprcoessing snippet from source code.

```
df1 = pd.read_csv('Full-Economic-News-DFE-
839861.csv',encoding='ISO-8859-1')
df2 = pd.read_csv('us-economic-newspaper.csv', encoding='ISO-
8859-1')
df = pd.concat([df1, df2], sort=False)
df = df.sample(frac=1).reset_index(drop=True)

# Data Preprocessing steps
# step 1; select columns "text", "positivity"
# step 2; remove rows with NaN positivity
# step 3; remove punctuation
# step 4; remove stop words
# step 5; stemming words, to get effictive part of word in
training
# step 6; count vectorizer, convert articles to vecotrs of
0's - 1's



def tweet_to_words(raw_tweet):
```

```
    letters_only = re.sub("[^a-zA-Z]", " ",raw_tweet)
    words = letters_only.lower().split()
    stops = set(stopwords.words("english"))
    meaningful_words = [w for w in words if not w in stops]
    return(" ".join( meaningful_words ))

Df = df[["positivity", "text"]].dropna()
Df["sentiment"] = 0
Df["words"] = ""
Df.sentiment = Df.positivity.apply(lambda x: 0 if x<5 else 1)
Df.words = Df.text.apply(lambda x: tweet_to_words(x))

del(Df['text'])
del(Df['positivity'])

train, test = train_test_split(Df,test_size=0.2
,random_state=42)
train, val = train_test_split(train,test_size=0.25
,random_state=42)

X_train, y_train = train.words, train.sentiment
X_test, y_test = test.words, test.sentiment
X_val, y_val = val.words, val.sentiment


def stemmming(x):
    porter = PorterStemmer()
    splitted = x.split(' ')
    x = ' '.join([porter.stem(w) for w in splitted])
    return x

X_train = X_train.apply(lambda x:stemmming(x))
X_test = X_test.apply(lambda x:stemmming(x))
X_val = X_val.apply(lambda x:stemmming(x))

v = CountVectorizer(analyzer = "word")
X_train_features = v.fit_transform(X_train)
X_test_features = v.transform(X_test)
X_val_features = v.transform(X_val)
v = CountVectorizer(analyzer = "word")

X = pd.concat([X_train, X_test, X_val])

X_features = v.fit_transform(X)

y = pd.concat([y_train, y_test, y_val])
```

2.2 Classifiers

While dealing this dataset, we had to use more than one classifier. Mainly, because wee seek a model with best possible performance. And because the models we've trained at first weren't well enough for our purpose or possible to launch.

What classifiers we used, they're mentioned following and ordered chronological.

Trained Classifiers

- Logistic Regression
- K-Nearest Neighbor
- Support Vector Classifier
- Decision Trees
- Random Forest

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import accuracy_score
from sklearn.model_selection import validation_curve
```

Every Classifier has what is called **'Hyper-parameter'**. It is, simply, some parameters that a classifier use in its equations and processing, or learning. So, In addition to training more than one model, we need to train that same model again with some modification, which mostly, changes its performance, increasing or decreasing or even same result are all possible results. Modifying or changing hyper-parameter values is called Hyper-paramter *tuning*.

Searching for best values for Hyper-parameters is, in some way, similar to searching for some value in, say array, or any other data structure. So, We may use some already implemented and ready-to-use algorithm, but there's also normal way for searching, which is, just like linear-search. Here we try every possible value for every Hyper-parameter and train this model using this value. But such method is too much power- and time-consuming, instead we try various values in a range this is known to give best results. It is, after all, known that Machine Learning Algorithms and Techniques are some form of research and there are already many researchers and papers for all known Machine Learning Algorithms.

Here, I used the normal method to search and is used with all classifiers and it's fairly simple. Iterating over set of values in some common range of some parameter. Using **'validation_curve'** function which automatically train the model over given (X, y) dataset with possible variations of only one parameter, we can compare between all models.

Following, we talk about the classifiers we used and what modifications we made.

2.2.1 Logistic Regression

Logistic Regression classifier has many hyper-parameters, and what arouse our interest, here, are **'max_iter'** and **'C'**.
Iterating over set of values of **'max_iter'**, we train models with varying value of another hyper-parameter, **'C'**, or the regularization term.

Following is the code, in Python, we used to train our Logistic Regression model.

```
def LogisticRegressionModel():
    best_acc = 0
    c_range = [.1, .3, 10, 30]#, 100]
    # for c in ([.01,.03,.1,.3,10]):#,3,10,30,100
    for itr in ([3,10,30,100,300,1000]):
        classifier =
LogisticRegression(solver='liblinear',max_iter=itr)
        prop = ' - itr ' + str(itr)
        title = classifier.__class__.__name__ + prop
        train_scores, test_scores = validation_curve(
```

```
              classifier, X_features, y, 'C', c_range, cv=10)
```

We trained our models, so we need to store performance for those trained models. Following, the code we use to store preformance values.

```
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
```

Now, we indeed have significant data that represent our performance for all models, but that data has little to no meaning, yet. We need to visualize this data to give some information about if this model is better than other models, or about some values of a hyper-parameters are better than other other values. Following, the code we used to plot, or visualize, our performance.

We use results from **'validation_curve'**, after the training is done, and it returns two parameters represent training, test accuracy respectively.
Each parameter is essentially a list of results, each element represent result of training at some variation of variated parameter, which is here, **'C'** and variations are **'c_range'** list.
Mean and std for training and testing results, give fair results across all models trained.

```
plt.figure(figsize=(8,5))
plt.xlabel("C values")
plt.ylabel("Score")
plt.title('Accuracy = {0:.05f}'.format(100*test_scores_mean.max()) + prop)
plt.grid()
lw = 2
plt.plot(c_range, train_scores_mean, 'o-', color="r",
         label="Training score", lw=lw)
plt.plot(c_range, test_scores_mean, 'o-', color="g",
         label="Test score", lw=lw)
plt.fill_between(n_range, train_scores_mean - train_scores_std,
        train_scores_mean+train_scores_std, alpha=0.1, color="r", lw=lw)
plt.fill_between(n_range, test_scores_mean - test_scores_std,
        test_scores_mean+test_scores_std, alpha=0.1, color="g", lw=lw)
plt.legend(loc="best")
best_acc = max(best_acc, test_scores_mean.max())
```

X-axis represents variations of variated parmeter, ‘C’. Y-axis represents performance on scale 0.0 – 1.0 , or 0 – 100%. In section 2.4, We see our results.

2.2.2 K-Nearest Neighbor

K-Nearest Neighbors Classifier, similarly, varying parameter ‘n_neighbors’, over n_range array. This parameter represents the number which is taken into consideration when deciding for some point if it is ‘near’ to another.
Iterating over set of values while varying ‘max_iter’, which represents maximum number the algorithm will train and change weights more and for better until convergence.

Following is the code, in Python, we used to train our Logistic Regression model.

```
def KNeighborsClassifierModel():
    classifier = KNeighborsClassifier(n_neighbors=5, leaf_size=3)
    n_range = range(3,7)
    prop = ''
    title = classifier.__class__.__name__ + prop
    train_scores, test_scores = validation_curve(
        KNeighborsClassifier(), X_train_features, y_train,
                               'n_neighbors', n_range, cv=10)
```

Again, plotting is useful when we need to know why accuracy isn’t what we expected, if it’s the case. And That’s because validation curve shows us how the model behaves through training and after many iterations across variations of some parameter, and that helps us to know if some solution may help or not. Because, we not only have model behavior at some state, but we know also how it’ll behave if we change some variable.

```
plt.figure(figsize=(8,5))
plt.xlabel("N-Neigbors")
plt.ylabel("Score")
plt.title('Accuracy = {0:.05f}'.format(100*test_scores_mean.max()) + prop)
plt.grid()
lw = 2
plt.plot(n_range, train_scores_mean, 'o-', color="r",
         label="Training score", lw=lw)
plt.plot(n_range, test_scores_mean, 'o-', color="g",
         label="Test score", lw=lw)
```

```
plt.fill_between(n_range, train_scores_mean - train_scores_std,
        train_scores_mean+train_scores_std, alpha=0.1, color="r", lw=lw)
plt.fill_between(n_range, test_scores_mean - test_scores_std,
        test_scores_mean+test_scores_std, alpha=0.1, color="g", lw=lw)
plt.legend(loc="best")
Accuracy.append(test_scores_mean.max())
```

2.2.3 Support Vector Classifier

Support Vector Classifier, similarly, varying parameters ‘kernel’, ‘degree’ over range 2-5, and ‘C’ over c_range array.
‘C’ parameter represents the factor that affects Regularization term.
‘kernel’ variable iterates over 4 different kernels. Kernel here refers to the ‘d’ variable iterates over a set of dimensions for the ‘polynomial’ kernel.
‘degree’ is enabled when kernel is set as polynomial. This parameter refers to degree of features.

```
def SVCModel():
    c_range = [.001, .003, .01, .03, .1, .3, 1, 3, 10]
    best_acc = 0
    for krnl in ['linear', 'rbf', 'poly', 'sigmoid']:
        f = 0
        for d in range(2,5):
            classifier = SVC(kernel='rbf', gamma='scale', degree = d,
                         probability = True, max_iter = 200)
            if krnl != 'poly':
                prop = ' - krnl ' + krnl
                if f == 1:
                    continue
                else:
                    pass
            else:
                prop = ' - krnl poly with d ' + str(d)
            title = classifier.__class__.__name__ + prop
            train_scores, test_scores = validation_curve(
              classifier, X_train_features, y_train, 'C', c_range, cv=3)
```

Variations of variated parameter, ‘C’, are stored in ‘c_range’ list. Again, we here calculate mean and std for return parameter to get an accurate estimation.
Accuracy for this classifier is maximum result across all iterations at different kernels and degrees for polynomial kernel.

```
plt.figure(figsize=(8,5))
plt.xlabel("C")
plt.ylabel("Score")
plt.title('Accuracy = {0:.05f}'.format(100*test_scores_mean.max()) + prop)
```

```
plt.grid()
lw = 2
plt.plot(c_range, train_scores_mean, 'o-', color="r",
         label="Training score", lw=lw)
plt.plot(c_range, test_scores_mean, 'o-', color="g",
         label="Test score", lw=lw)
plt.fill_between(n_range, train_scores_mean - train_scores_std,
        train_scores_mean+train_scores_std, alpha=0.1, color="r", lw=lw)
plt.fill_between(n_range, test_scores_mean - test_scores_std,
        test_scores_mean+test_scores_std, alpha=0.1, color="g", lw=lw)
plt.legend(loc="best")
best_acc = max(best_acc, test_scores_mean.max())
f = 1
Accuracy.append(best_acc)
```

X-axis represents variations of variated parameter, ‘C’. Y-axis represents scoring on scale 0.0 – 1.0 , or 0 – 100%.
Foremost, What we may need from a validation curve, other than know about dataset training behavior, is to know how much this model is ‘good’. So, we may compare it to another model, or same one with different values for hyper parameters.

2.2.4 Decision Trees

Decision Tree Classifier, similarly, has hyper-parameters needs to be ‘tuned’.
‘crit’ variable iterates over 2 different criterion, either ‘gini’ or ‘entropy’, and what they refer to is how the classifier will behave in calculations and decisions when splitting data set. Both are useful and have their respective situations where they do best.

```
def DecisionTreeClassifierModel():
    s_range = [2,3,4,5,6]
    best_acc = 0
    for crit in ['gini', 'entropy']:
        classifier = DecisionTreeClassifier(criterion = crit)
        prop = ' - ' + crit
        title = classifier.__class__.__name__ + prop
        train_scores, test_scores = validation_curve(
                        classifier, X_train_features, y_train,
                        'min_samples_split', s_range, cv=10)
```

Variations of variated parameter, ‘min_samples_split’, are stored in ‘s_range’ list.
Again, we here calculate mean and std for return parameter to get an accurate estimation.

Accuracy for this classifier is maximum result between model based on ‘gini’ criterion or ‘entropy’.
X-axis represents variations of variated parameter, ‘min_samples_split’. Y-axis represents scoring on scale 0.0 – 1.0 , or 0 – 100%.

```
plt.figure(figsize=(8,5))
plt.xlabel("sample split")
plt.ylabel("Score")
plt.title('Accuracy = {0:.05f}'.format(100*test_scores_mean.max()) + prop)
plt.grid()
lw = 2
plt.plot(s_range, train_scores_mean, 'o-', color="r",
         label="Training score", lw=lw)
plt.plot(s_range, test_scores_mean, 'o-', color="g",
         label="Test score", lw=lw)
plt.fill_between(n_range, train_scores_mean - train_scores_std,
        train_scores_mean+train_scores_std, alpha=0.1, color="r", lw=lw)
plt.fill_between(n_range, test_scores_mean - test_scores_std,
        test_scores_mean+test_scores_std, alpha=0.1, color="g", lw=lw)
plt.legend(loc="best")
best_acc = max(best_acc, test_scores_mean.max())
Accuracy.append(best_acc)
```

2.2.5 Random Forest

Random Forest Classifier, similarly, has hyper-parameters needs to be ‘tuned’.
‘crit’ variable iterates over 2 different criterion, either ‘gini’ or ‘entropy’.
‘n_est’ variable iterates over a set of choices for ‘# neighbors’ what decision will be made based on.
‘s_range’ variable is a list represents variations for function to create validation curve.

```
def RandomForestClassifierModel():
  best_acc = 0
  s_range = [2,3,4,5,6]
  for crit in ['gini', 'entropy']:
    for n_est in [5,10,25,100,130,300]:
      print(crit, n_est)
      classifier = RandomForestClassifier(criterion=crit, n_estimators=n_est)
      prop =  ' - criterion = ' + crit + ' estimators = ' + str(n_est)
      title = classifier.__class__.__name__ + prop
      train_scores, test_scores = validation_curve(classifier,
                X_train_features, y_train, 'min_samples_split', s_range, cv=10)
```

Variations of variated parameter, ‘min_samples_split’, are stored in ‘s_range’ list.
Again, we here calculate mean and std for return parameter to get an accurate estimation. Accuracy for this classifier is maximum result across models based on ‘gini’ criterion or ‘entropy’, and #neighbors as ‘n_est’ variable.

X-axis represents variations of variated parameter, ‘min_samples_split’. Y-axis represents scoring on scale 0.0 – 1.0 , or 0 – 100%.

```
plt.figure(figsize=(8,5))
plt.xlabel("sample split")
plt.ylabel("Score")
plt.title('Accuracy = {0:.05f}'.format(100*test_scores_mean.max()) + prop)
plt.grid()
lw = 2
plt.plot(s_range, train_scores_mean, 'o-', color="r",
         label="Training score", lw=lw)
plt.plot(s_range, test_scores_mean, 'o-', color="g",
         label="Test score", lw=lw)
plt.fill_between(n_range, train_scores_mean - train_scores_std,
        train_scores_mean+train_scores_std, alpha=0.1, color="r", lw=lw)
plt.fill_between(n_range, test_scores_mean - test_scores_std,
        test_scores_mean+test_scores_std, alpha=0.1, color="g", lw=lw)
plt.legend(loc="best")
best_acc = max(best_acc, test_scores_mean.max())
Accuracy.append(best_acc)
```

2.3 Data Visualization

Following figures are training results, specifically Validation Curves and are called plots.
Every group of plots are enlisted under some classifier, and each plot represents some variations as mentioned below it.

2.3.1 Logistic Regression Classifier

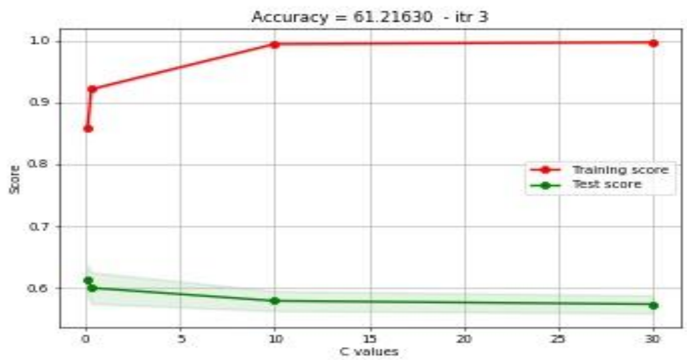


Figure 2.3.1.1 Logistic Regressin - maxItr 3

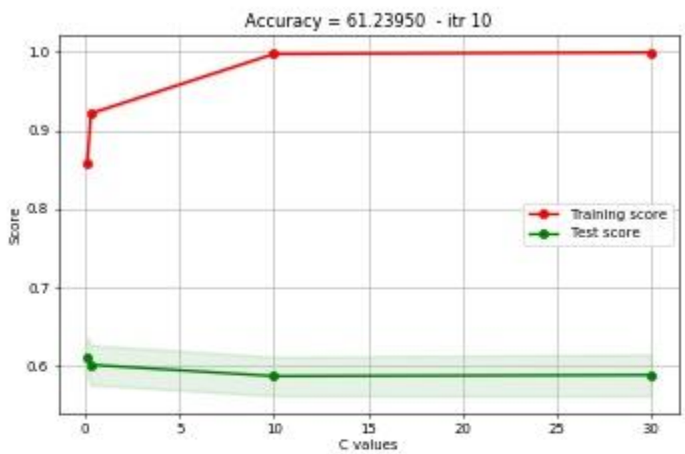


Figure 2.3.1.2 Logistic Regressin - maxItr 10

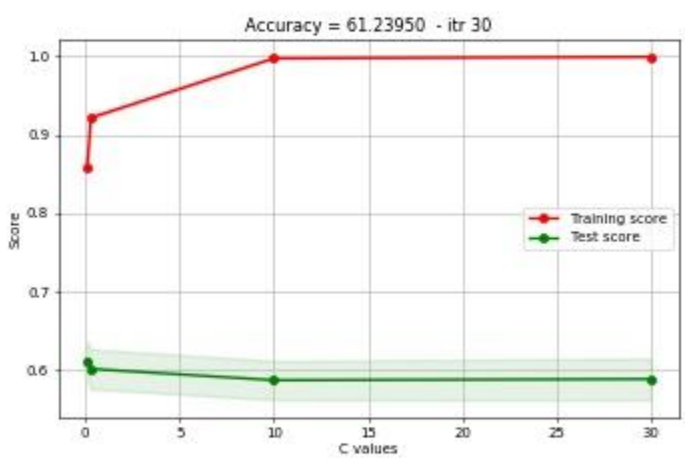


Figure 2.3.1.3 Logistic Regressin - maxItr 30

2.3.2 K-Nearest Neighbor Classifier

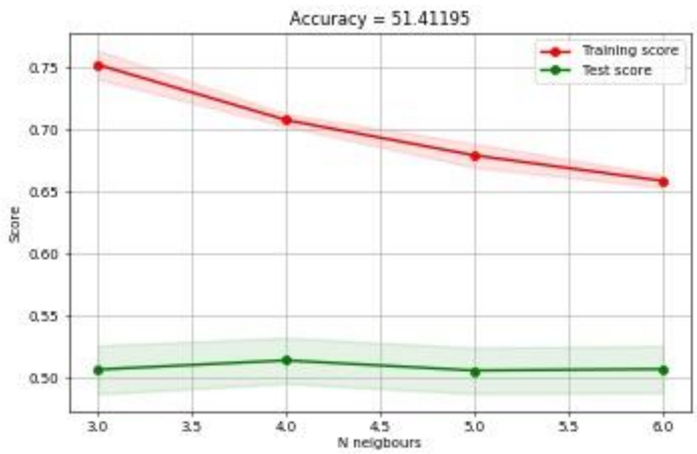


Figure 2.3.2.1 K-Nearest Neighbour - maxItr 30

2.3.3 Suppor Vector Classifier(SVC)

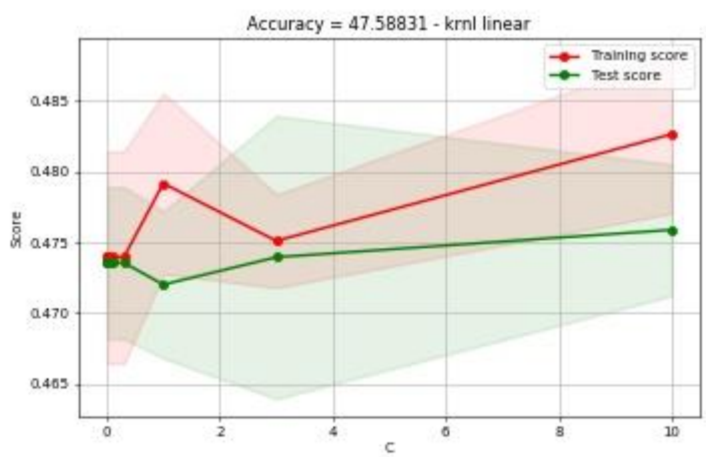


Figure 2.3.3.1 SVC with Linear kernel

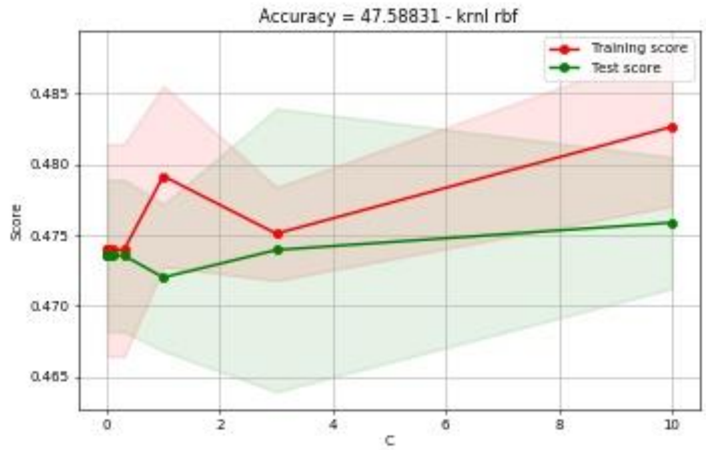


Figure 2.3.3.2 SVC with Radial Basis function kernel

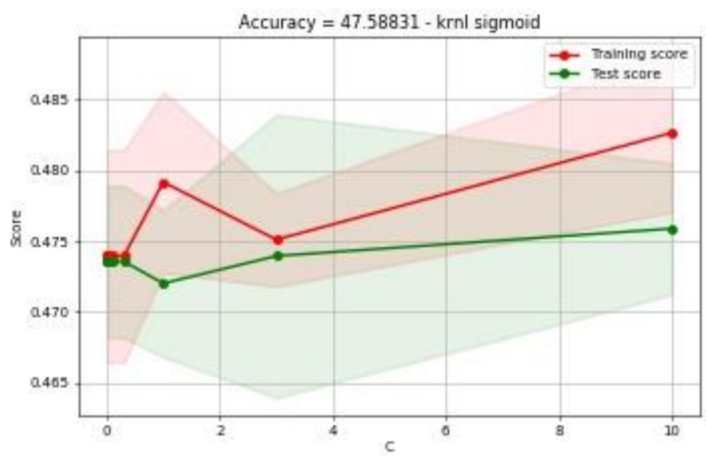


Figure 2.3.3.3 SVC with sigmoid\/gasussian kernel

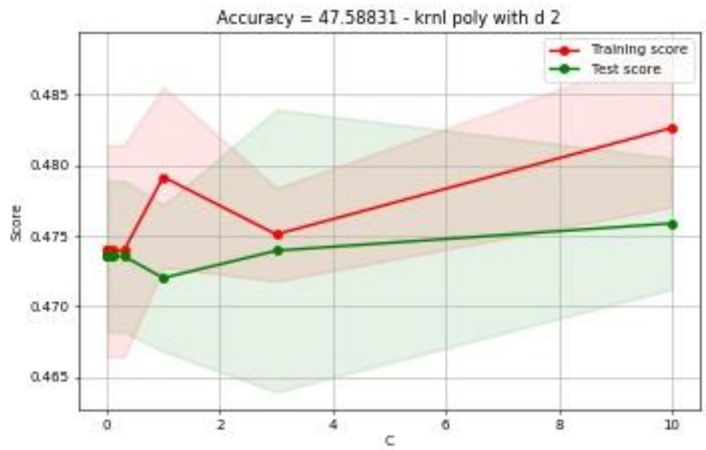


Figure 2.3.3.4 SVC with Polynomial kernel - degree 2

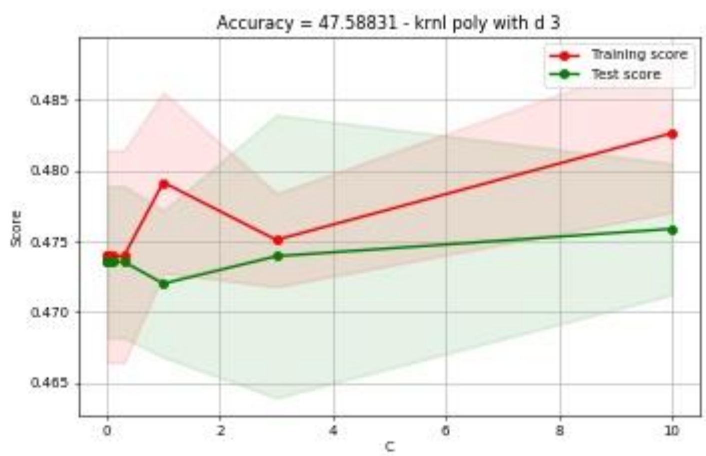


Figure 2.3.3.5 SVC with Polynomial kernel - degree 3

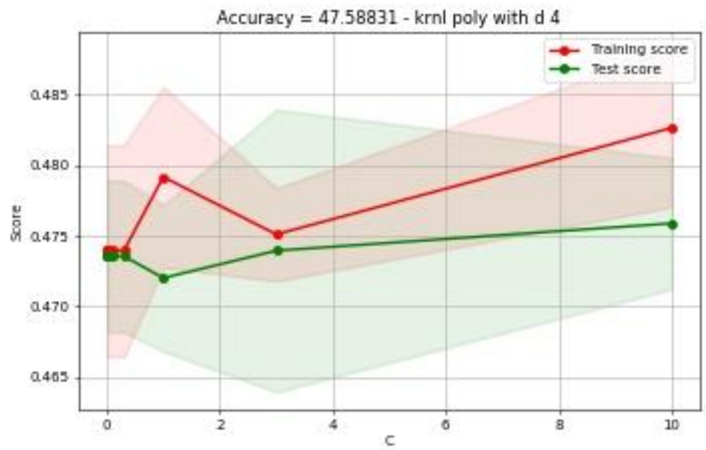


Figure 2.3.3.6 SVC with Polynomial kernel - degree 4

2.3.4 Decision Tree Classifier

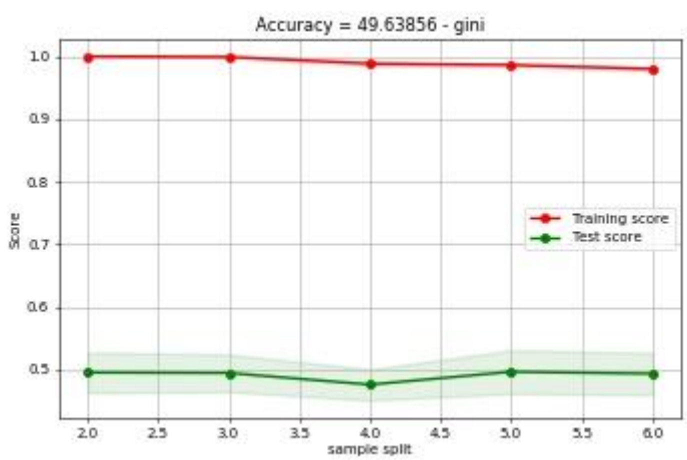


Figure 2.3.4.1 Decision Tree Classifier using gini

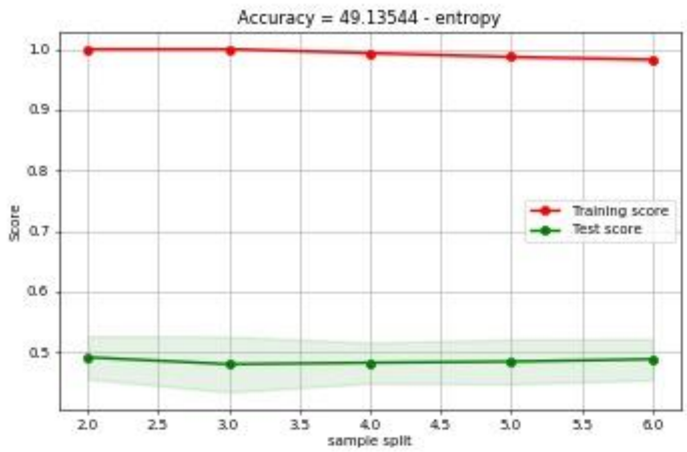


Figure 2.3.4.2 Decision Tree Classifier using entrop

2.3.5 Random Forest

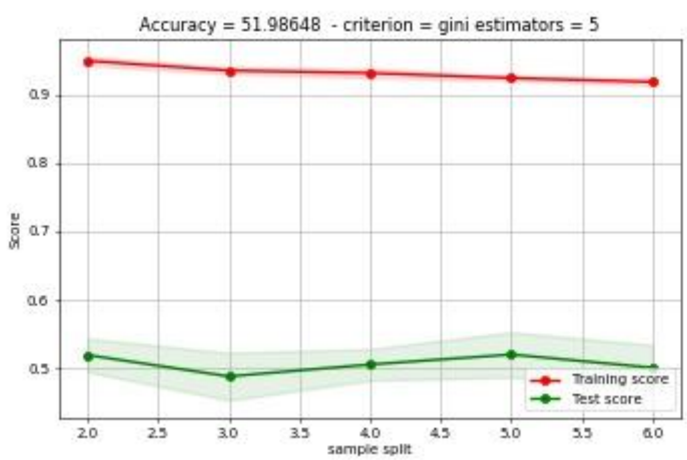


Figure 2.3.5.1 Random Forest Classifier
using gini - using 5 estimators

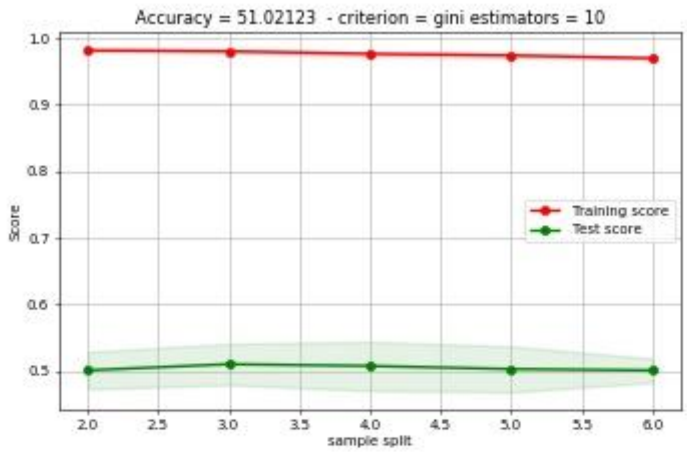


Figure 2.3.5.2 Random Forest Classifier
using gini - using 10 estimators

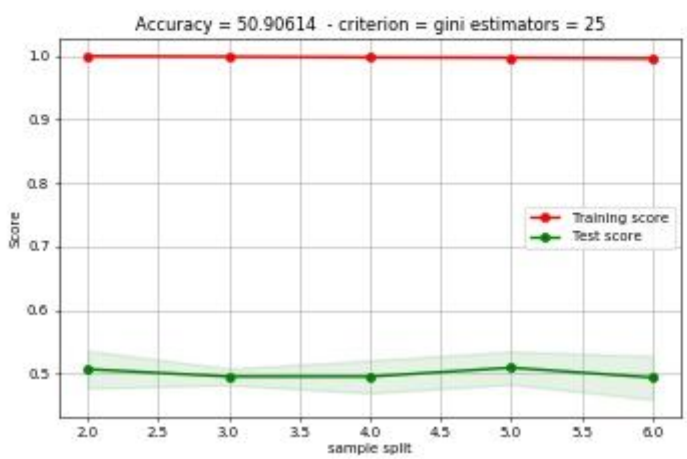


Figure 2.3.5.3 Random Forest Classifier
using gini - using 25 estimators

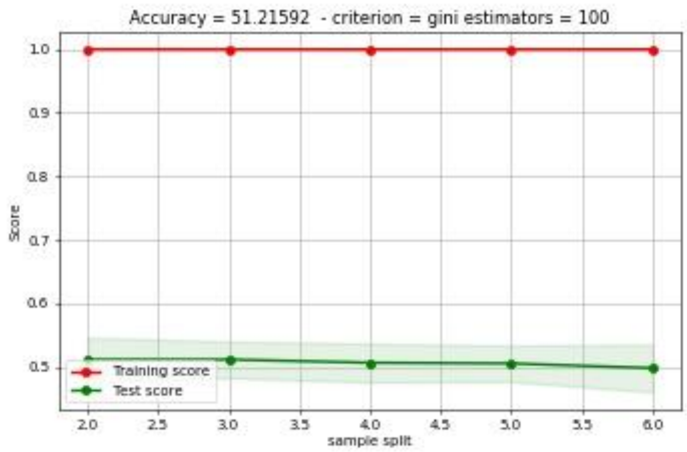


Figure 2.3.5.4 Random Forest Classifier
using gini - using 100 estimators

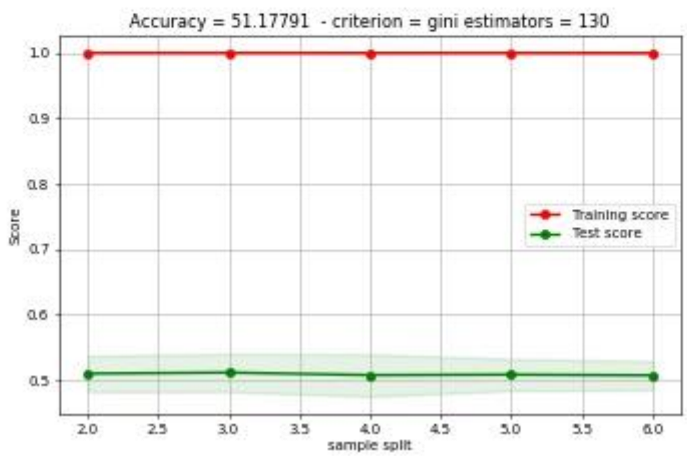


Figure 2.3.5.5 Random Forest Classifier
using gini - using 130 estimators

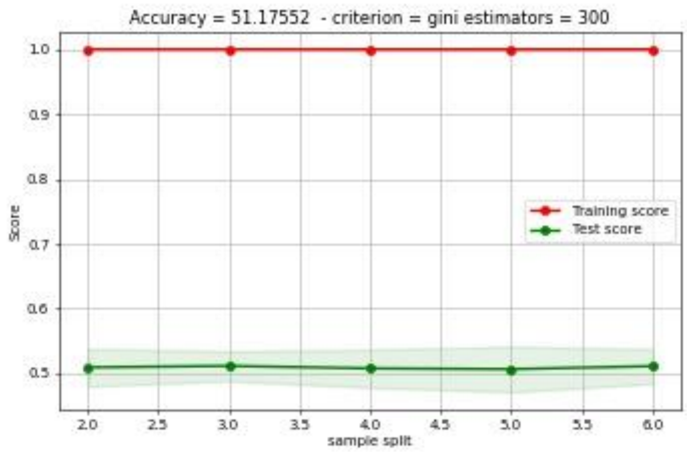


Figure 2.3.5.6 Random Forest Classifier
using gini - using 300 estimators

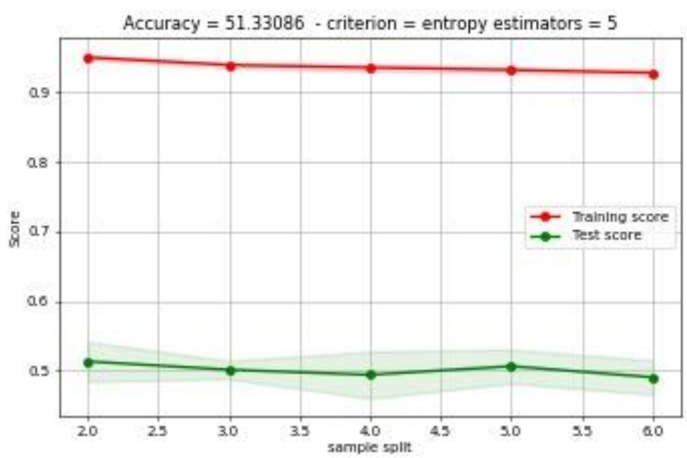


Figure 2.3.5.7 Random Forest Classifier
using entropy - using 5 estimators

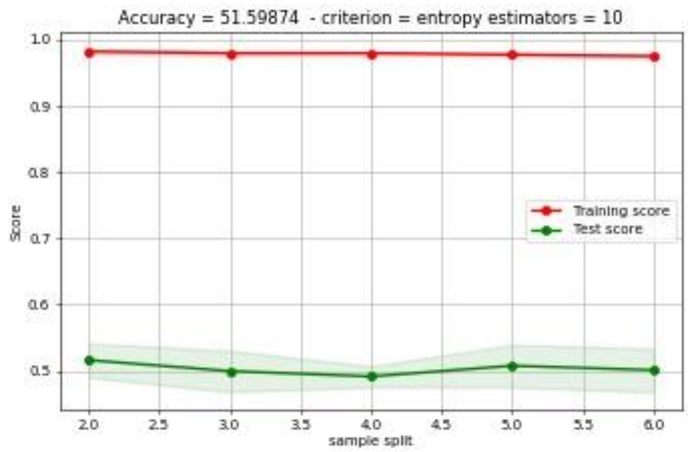


Figure 2.3.5.8 Random Forest Classifier
using entropy - using 10 estimators

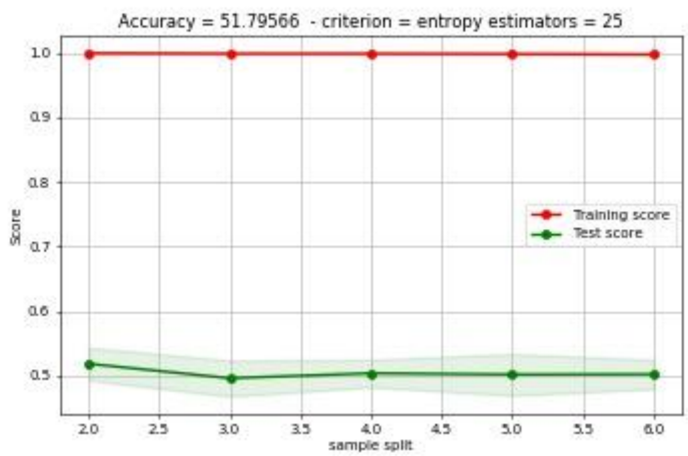


Figure 2.3.5.9 Random Forest Classifier
using entropy - using 25 estimators

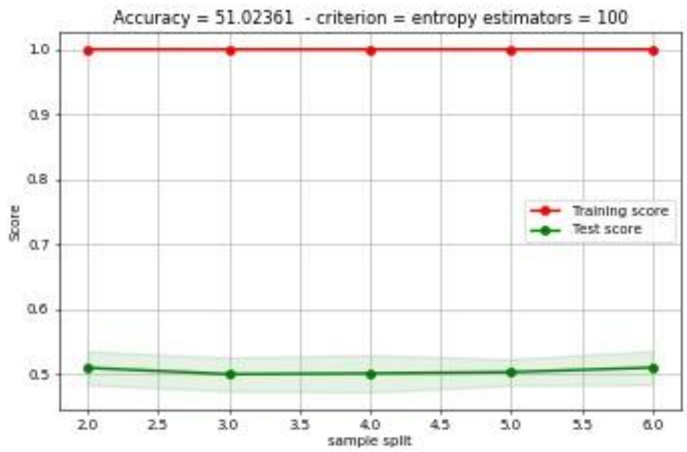


Figure 2.3.5.10 Random Forest Classifier
using entropy - using 100 estimators

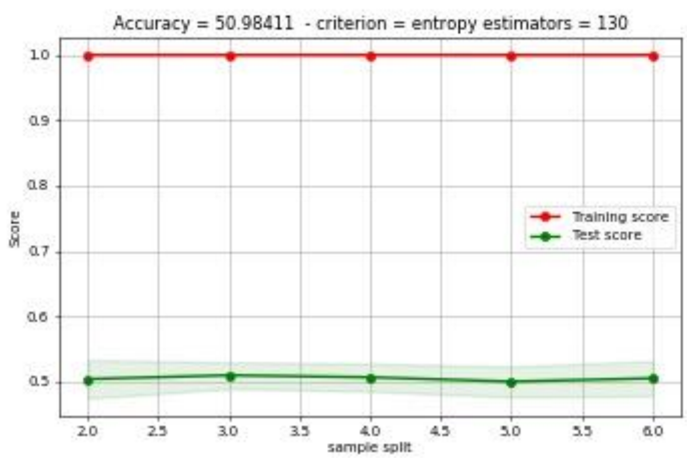


Figure 2.3.5.11 Random Forest Classifier
using entropy - using 130 estimators

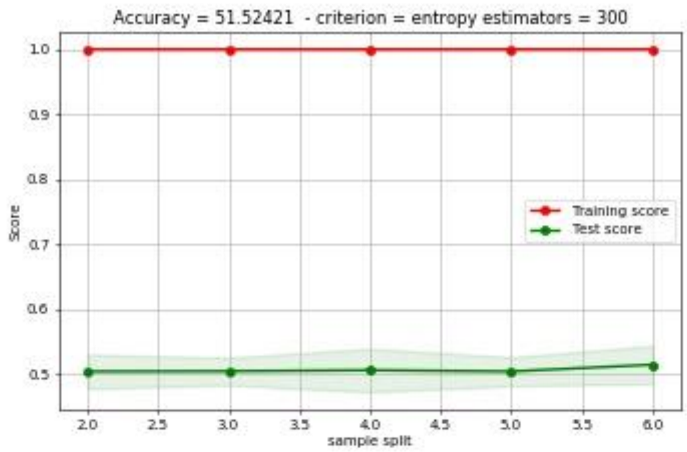


Figure 2.3.5.12 Random Forest Classifier
using entropy - using 300 estimators

2.4 Conclusion

**So far, our best model has accuracy of about 61.23%
Logistic Regression model, for this dataset, wins.**

Now, we've seen the data, knew about it and we even made a model that trains this data and generalizes how it deal with other articles or sentences and give a correct sentiment if it's positive or negative.

We may do one thing more, maybe we classify it as multiclassification [negative, neutral and positive], and such classification exists and is already common.

We also need to look into the data, see what words were the most important and the classifier give most attention, this way we may choose less features and prevent overfitting, which is a thing our model suffer due to insufficient data.

We also may train these models more on better CPU or using GPU, it'll help us search on 2 parameters and get better validation curves, and have a wider range of variations to test our model on.

One thing worth mentioning is that there're classifiers that are able to classify articles based on not only sentiment, but also subjective. It means that it can tell if this article relates to sport, economic, politics, medicine, etc. etc.

Chapter 3

Software Project Management Plan

3.1 Introduction

3.1.1 Project Overview

Motivation:

Number of online articles is huge and increases over time. Sometimes users need positive/negative opinions about specific subject in economy so they spend time and effort for reading all available articles online to know the positivity/negativity about it.

Project Objective:

We build this project to facilitate the process of articles revisions and spot a critical opinion about each one.

3.1.2 Project Deliverables

Articles Blog is a website for:

– Writing and Analyzing articles about Economy.
– Finding positive / negative classified articles based on Natural language processing.

3.2 Project Organization

3.2.1 Software Process Model

A (software/system) process model is a description of the sequence of activities carried out in SE project, and the relative order of these activities.

It has many types such as:

Waterfall Model

The waterfall model is a sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order. In the waterfall model, you must plan and schedule all of the activities before starting working on them (plan-driven process). The phases of the waterfall model are: Requirements, Design, Implementation, Testing, and Maintenance.

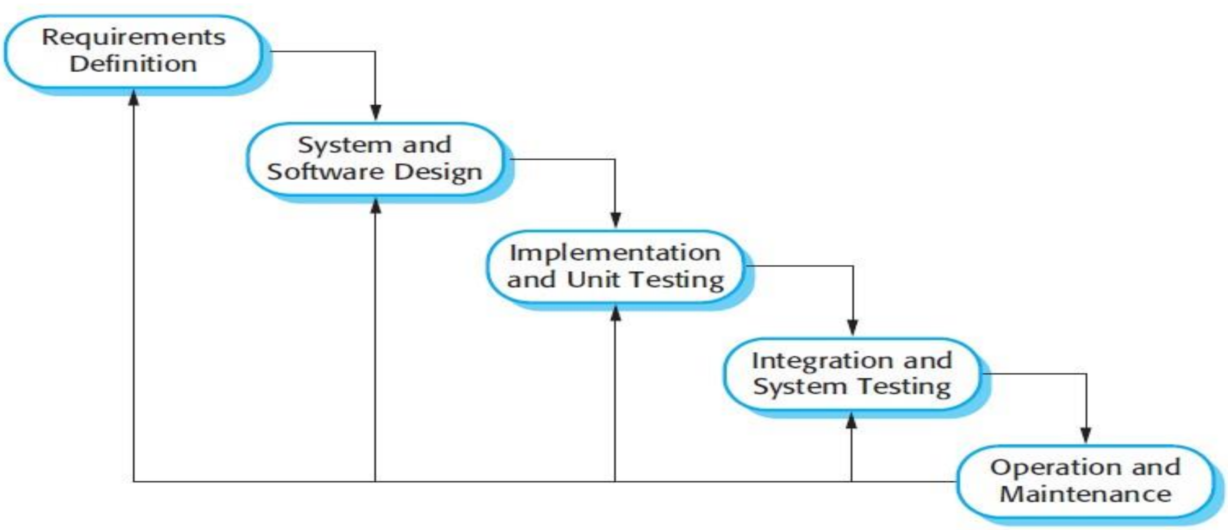


Figure 3.1 Waterfall Model

The Nature of Waterfall Phases

In principle, the result of each phase is one or more documents that should be approved, and the next phase shouldn’t be started until the previous phase has completely been finished.

In practice, however, these phases overlap and feed information to each other. For example, during design, problems with requirements can be identified, and during coding, some of the design problems can be found, etc.

The software process therefore is not a simple linear but involves feedback from one phase to another. So, documents produced in each phase may then have to be modified to reflect the changes made.

When to Use?

In principle, the waterfall model should only be applied when requirements are well understood and unlikely to change radically during development as this model has a relatively rigid structure which makes it relatively hard to accommodate change when the process in underway.

Advantages of waterfall model:

– It allows for departmentalization and managerial control.
– Simple and easy to understand and use.
– Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
– Phases are processed and completed one at a time.
– Works well for smaller projects where requirements are very well understood.
– A schedule can be set with deadlines for each stage of development and a product can proceed through the

development process like a car in a car-wash, and theoretically, be delivered on time.

Disadvantages of waterfall model:

– It does not allow for much reflection or revision.
– Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
– No working software is produced until late during the life cycle.

V-Model

V-Model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. V-Model is one of the many software development models.
Testing of the product is planned in parallel with a corresponding phase of development in V-model.

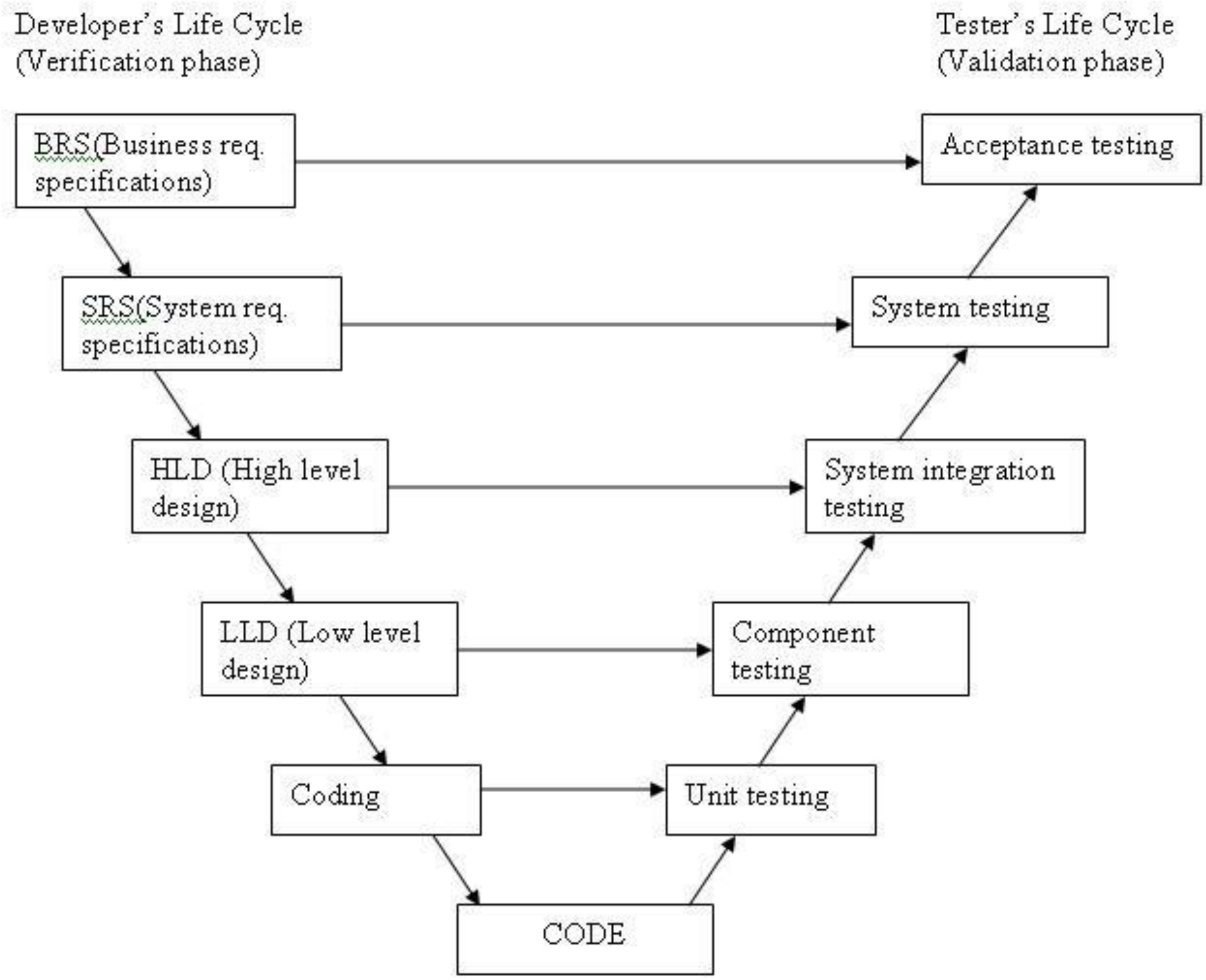


Figure 3.2 V-model

The various phases of the V-model are as follows:

Requirements like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase focuses on system architecture and design. It provides overview of solution, platform, system, product and service/process. An integration test plan is created in this phase as well to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component test are created in this phase as well.

The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now use. Coding: This is at the bottom of the V-Shape model. Module design is converted into code by developers. Unit testing is performed by the developers on the code written by them.

When to use the V-model:

– The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
– The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.
– High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

Advantages of V-model:

– Simple and easy to use.
– Testing activities like planning, test designing happens well before coding. This saves a lot of time, hence higher chance of success over the waterfall model.
– Proactive defect tracking – that is defects are found at early stage.
– Avoids the downward flow of the defects.
– Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

– Very rigid and least flexible.
– Software is developed during the implementation phase, so no early prototypes of the software are produced.
– If any changes happen in midway, then the test documents along with requirement documents must be updated.

Prototype Model

The basic idea in Prototype model is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a software development model. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:

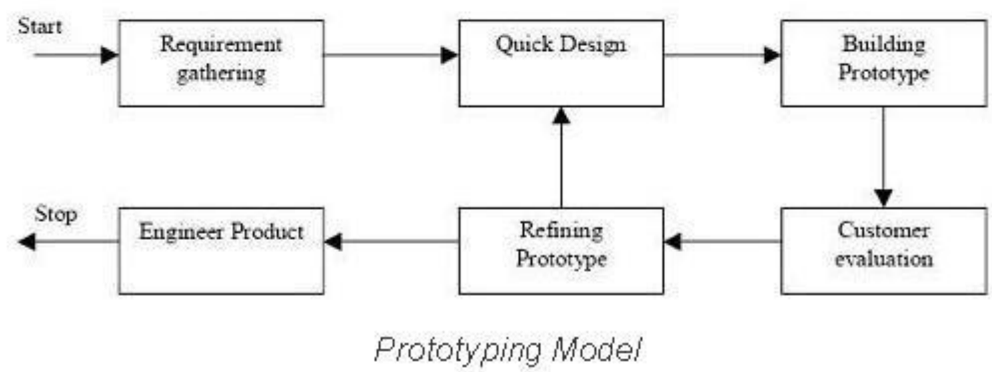


Figure 3.3 Prototype model

Advantages of Prototype model:

– Users are actively involved in the development.
– Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
– Errors can be detected much earlier.
– Quicker user feedback is available leading to better solutions.
– Missing functionality can be identified easily.
– Confusing or difficult functions can be identified.

Disadvantages of Prototype model:

– Leads to implementing and then repairing way of building systems.
– Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
– Incomplete application may cause application not to be used as the full system was designed.
– Incomplete or inadequate problem analysis.

3.2.2 Tools and Techniques

In this section, we mention the tools and techniques that we have used to implement our project like programming languages.

Language

We use Python. Why? The language is easy and widespread and has many libraries.

Libraries

– re
– nltk
– pandas
– stopwords from nltk.corpus
– word_tokenize from nltk.tokenize
– PorterStemmer from nltk.stem.porter

Training Classifiers

– Logistic Regression
– K Nearest Neighbors
– Support Vector Classifier
– Decision Tree
– Random Forests

DataBases

MYSQL

– MySQL is a very popular, open source DBMS
– MySQL databases are relational
– Officially pronounced “my Ess Que Ell” (not my sequel).
– Handles very large databases;
– very fast performance; reliable.
– MySQL is compatible with standard SQL
– Why are we using MySQL?
– Free (much cheaper than Oracle!)
– Each student can install MySQL locally.
– Multi-user access to a number of databases offered
– Easy to use Shell for creating tables, querying tables, etc.
– Easy to use with Java JDBC
– MySQL is frequently used by PHP and Perl
– Commercial version of MySQL is also provided (including technical support)

MONGODB

– MySQL is a very popular, open source RDBMS
– MySQL databases are NON relational
– Handles very large databases;

– very fast performance; reliable.
– Why are we using MongoDB?
– Free
– Each student can install it locally.
– Multi-user access to a number of databases offered
– Easy to use Shell for creating collections, querying collections, etc.
– MONGODB is frequently used by PHP and Perl

BackEnd (Django PlatForm)

It is python based platform for web development

Loose coupling

A fundamental goal of Django's stack is loose coupling and tight cohesion. The various layers of the framework shouldn't "know" about each other unless absolutely necessary.
For example, the template system knows nothing about Web requests, the database layer knows nothing about data display and the view system doesn't care which template system a programmer uses. Although Django comes with a full stack for convenience, the pieces of the stack are independent of another wherever possible.

Less code

Django apps should use as little code as possible; they should lack boilerplate. Django should take full advantage of Python's dynamic capabilities, such as introspection.

Quick development

The point of a Web framework in the 21st century is to make the tedious aspects of Web development fast. Django should allow for incredibly quick Web development.

Don’t repeat yourself (DRY)

Every distinct concept and/or piece of data should live in one, and only one, place. Redundancy is bad. Normalization is good. The framework, within reason, should deduce as much as possible from as little as possible.

Explicit is better than implicit

This is a core Python principle listed in PEP 20, and it means Django shouldn’t do too much “magic.” Magic shouldn’t happen unless there’s a really good reason for it. Magic is worth using only if it creates a huge convenience unattainable in other ways, and it isn’t implemented in a way that confuses developers who are trying to learn how to use the feature.

Consistency

The framework should be consistent at all levels. Consistency applies to everything from low-level (the Python coding style used) to high-level (the “experience” of using Django).

3.3 Project Management Plans

Dependencies and Constraints:

We were depending on the economic online newspaper articles between 1951-2014ad As there was a constraint that the used data must be real to build an accurate system model with a good performance.

Risks and Contingencies:

There were some risks as there was no suitable machine/hardware for data training and data scarcity.

Chapter 4

Analysis & Design

4.1 Design

4.1.1 ERD

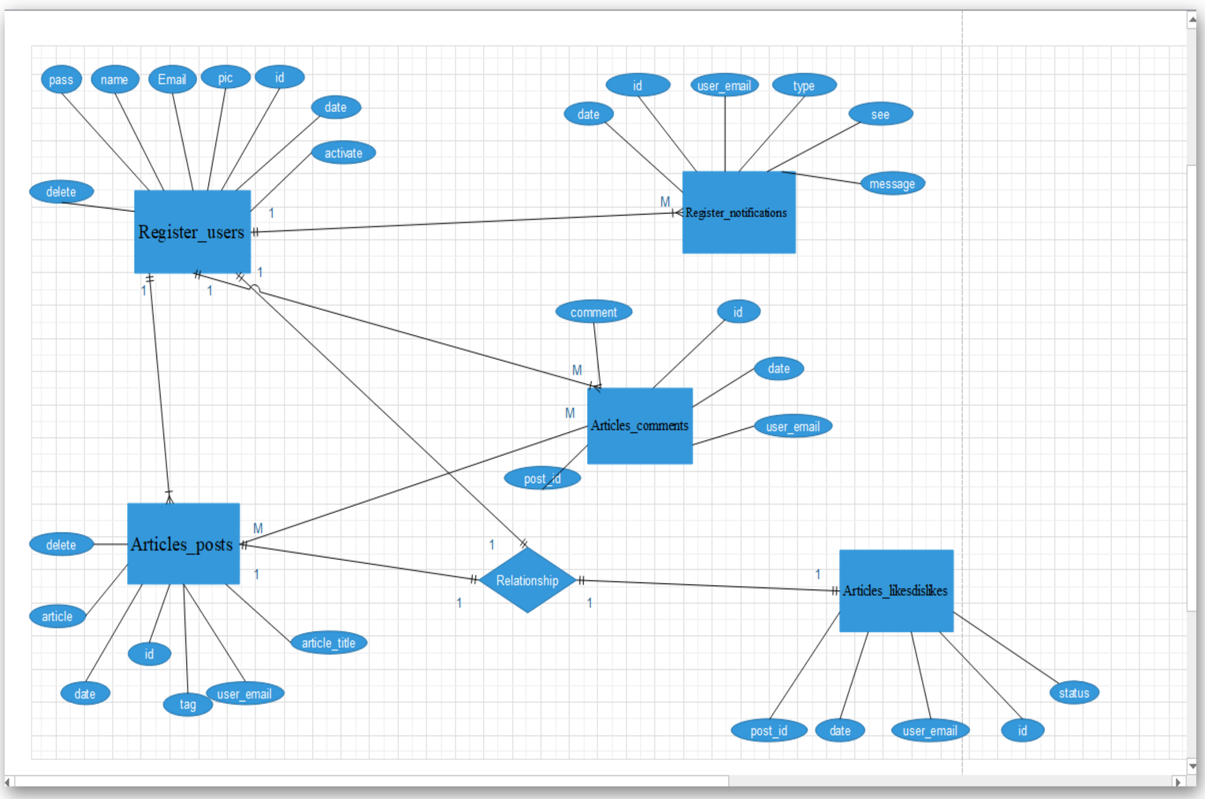
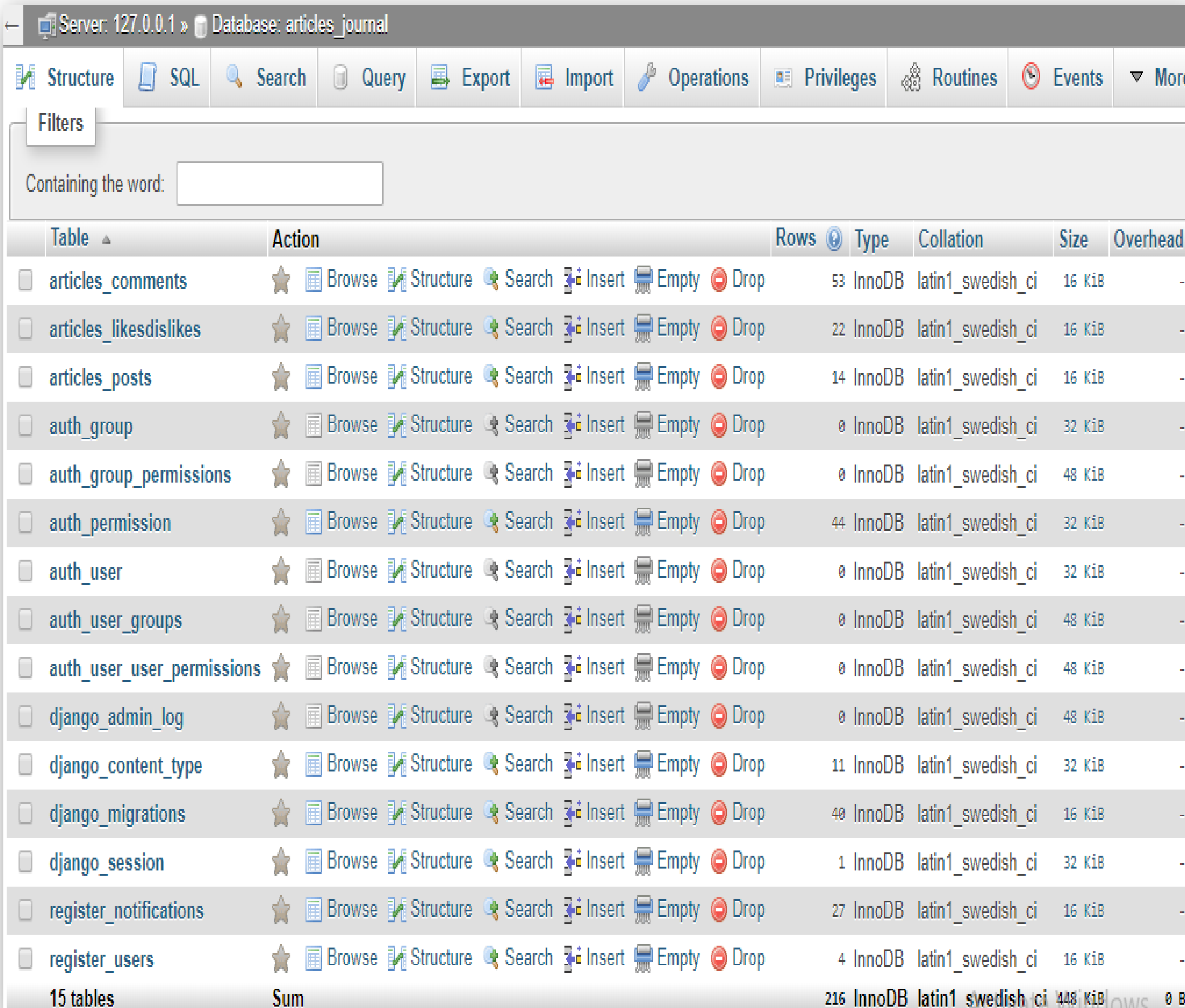


Figure 4.0 ERD

4.1.2 Database tables

1. Database Name : “articles_journal”

Command :”CREATE DATABASE IS NOT EXISTS articles_journal`”;


Server: 127.0.0.1 » Database: articles_journal

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | More

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
articles_comments	Browse Structure Search Insert Empty Drop	53	InnoDB	latin1_swedish_ci	16 KiB	-
articles_likesdislikes	Browse Structure Search Insert Empty Drop	22	InnoDB	latin1_swedish_ci	16 KiB	-
articles_posts	Browse Structure Search Insert Empty Drop	14	InnoDB	latin1_swedish_ci	16 KiB	-
auth_group	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
auth_group_permissions	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
auth_permission	Browse Structure Search Insert Empty Drop	44	InnoDB	latin1_swedish_ci	32 KiB	-
auth_user	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
auth_user_groups	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
auth_user_user_permissions	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
django_admin_log	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
django_content_type	Browse Structure Search Insert Empty Drop	11	InnoDB	latin1_swedish_ci	32 KiB	-
django_migrations	Browse Structure Search Insert Empty Drop	40	InnoDB	latin1_swedish_ci	16 KiB	-
django_session	Browse Structure Search Insert Empty Drop	1	InnoDB	latin1_swedish_ci	32 KiB	-
register_notifications	Browse Structure Search Insert Empty Drop	27	InnoDB	latin1_swedish_ci	16 KiB	-
register_users	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	16 KiB	-
15 tables	Sum	216	InnoDB	latin1_swedish_ci	448 KiB	0 B

Figure 4.1 Database articles_journal

2. Tables names and discerption
   1. Register_users:-
      Discerption :here, we store users data

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	Name	varchar(40)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	Email	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	Date	datetime(6)			No	None			Change Drop Primary Unique Index More
5	Picture	varchar(40)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
6	Password	varchar(40)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
7	Activate	tinyint(1)			No	None			Change Drop Primary Unique Index More
8	Deleted	tinyint(1)			No	None			Change Drop Primary Unique Index More

Figure 4.2 Register_Users

Command:

“CREATE TABLE IF NOT EXISTS `register_users`(
`id` INT(11) NOT NULL AUTO_INCREAMENT PRIMARY KEY,
`Name` VARCHAR(400) NOT NULL,
`Email` VARCHAR(100) NOT NULL,
`Date` DATETIME(6) NOT NULL,
`Picture` VARCHAR(40) NOT NULL,
`Password` VARCHAR(40) NOT NULL,
`Activate` TINYINT(1) NOT NULL,
`Deleted` TINYINT(1) NOT NULL,
)ENGINE=InnoDB DEFAULT CHARSET=utf8
AUTO_INCREAMENT=1;

2. Register_notifications

   Discerption : here, we store users notifications

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	User_Email	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	Type	int(11)			No	None			Change Drop Primary Unique Index More
4	Message	varchar(300)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
5	Date	datetime(6)			No	None			Change Drop Primary Unique Index More
6	See	tinyint(1)			No	None			Change Drop Primary Unique Index More

Figure 4.3 Register_notifications

Command:

```
CREATE TABLE IF NOT EXISTS `register_notifications`(

id` INT(11) NOT NULL AUTO_INCREAMENT
 PRIMARYKEY,
 `User_Email` VARCHAR(100) NOT NULL,
 `Type` INT(11) NOT NULL,
 `Message` VARCHAR(300) NOT NULL,
 `Date` DATETIME(6) NOT NULL,
 `See` TINYINT(1) NOT NULL,
 )ENGINE=InnoDB DEFAULT CHARSET=utf8
 AUTO_INCREAMENT=1;
```

3. Articles_posts

Discerption : here, we store users posts

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
	1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
	2	User_Email	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
	3	ArticleTitle	varchar(50)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
	4	Article	varchar(2000)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
	5	Deleted	tinyint(1)			No	None			Change Drop Primary Unique Index More
	6	Date	datetime(6)			No	None			Change Drop Primary Unique Index More
	7	Tags	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More

Figure 4.4 Articles_posts

Command:

CREATE TABLE IF NOT EXISTS `register_notifications`(
`id` INT(11) NOT NULL AUTO_INCREAMENT PRIMARY KEY,
`User_Email` VARCHAR(100) NOT NULL,
`ArticleTitle` VARCHAR(50) NOT NULL,
`Article` VARCHAR(2000) NOT NULL,
`Deleted` TINYINT(1) NOT NULL,
`Date` DATETIME(6) NOT NULL,
`Tags` VARCHAR(100) NOT NULL,
)ENGINE=InnoDB DEFAULT CHARSET=utf8
AUTO_INCREAMENT=1;

4. Articles_likesdislikes

Discerption : here, we store users likes and dislikes in specific articles.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	User_Email	varchar(110)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	Post_id	varchar(11)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	Date	datetime(6)			No	None			Change Drop Primary Unique Index More
5	Status	tinyint(1)			No	None			Change Drop Primary Unique Index More

Figure 4.5 Articles_likesdislikes

Command:

```
CREATE TABLE IF NOT EXISTS `register_notifications`(
`id` INT(11) NOT NULL AUTO_INCREAMENT PRIMARY KEY,
`User_Email` VARCHAR(100) NOT NULL,
`Post_id` VARCHAR(11) NOT NULL,
`Date` DATETIME(6) NOT NULL,
`Status` TINYINT(1) NOT NULL,
)ENGINE=InnoDB DEFAULT CHARSET=utf8
AUTO_INCREAMENT=1;
```

5. Articles_comments

Discerption:- here, we store users comments in specific articles

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	Post_id	int(11)			No	None			Change Drop Primary Unique Index More
3	User_Email	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
4	Comment	varchar(500)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
5	Date	datetime(6)			No	None			Change Drop Primary Unique Index More

Figure 4.6 Articles_comments

Command:

```
CREATE TABLE IF NOT EXISTS `register_notifications`(
 `id` INT(11) NOT NULL AUTO_INCREAMENT PRIMARY KEY,
`User_Email` VARCHAR(100) NOT NULL,
 `Post_id` VARCHAR(11) NOT NULL,
`Comment` VARCHAR(500) NOT NULL,
`Date` DATETIME(6) NOT NULL,
)ENGINE=InnoDB DEFAULT CHARSET=utf8
AUTO_INCREAMENT=1;
```

6. Platform Tables

1. django_session

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	session_key	varchar(40)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index Spatial More
2	session_data	longtext	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index Spatial More
3	expire_date	datetime(6)			No	*None*			Change Drop Primary Unique Index Spatial More

Figure 4.7 Django_session

2. django_migrations

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	app	varchar(255)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index More
3	name	varchar(255)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index More
4	applied	datetime(6)			No	*None*			Change Drop Primary Unique Index More

Figure 4.8 Django_migrations

3. django_content_type

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	app_label	varchar(100)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index More
3	model	varchar(100)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index More

Figure 4.9 Django_content_type

4. django_admin_log

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
	1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique More
	2	action_time	datetime(6)			No	*None*			Change Drop Primary Unique More
	3	object_id	longtext	latin1_swedish_ci		Yes	*NULL*			Change Drop Primary Unique More
	4	object_repr	varchar(200)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique More
	5	action_flag	smallint(5)		UNSIGNED	No	*None*			Change Drop Primary Unique More
	6	change_message	longtext	latin1_swedish_ci		No	*None*			Change Drop Primary Unique More
	7	content_type_id	int(11)			Yes	*NULL*			Change Drop Primary Unique More
	8	user_id	int(11)			No	*None*			Change Drop Primary Unique More

Figure 4.10 Django_admin_log

5. auth_user_user_permissions

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
	1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique Index Spatial More
	2	user_id	int(11)			No	*None*			Change Drop Primary Unique Index Spatial More
	3	permission_id	int(11)			No	*None*			Change Drop Primary Unique Index Spatial More

Figure 4.11 Auth_user_user_permissions

6. auth_user_groups

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
	1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique Index Spatial More
	2	user_id	int(11)			No	*None*			Change Drop Primary Unique Index Spatial More
	3	group_id	int(11)			No	*None*			Change Drop Primary Unique Index Spatial More

Figure 4.12 Auth_user_groups

7. auth_user

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	password	varchar(128)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	last_login	datetime(6)			Yes	NULL			Change Drop Primary Unique More
4	is_superuser	tinyint(1)			No	None			Change Drop Primary Unique More
5	username	varchar(150)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
6	first_name	varchar(30)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
7	last_name	varchar(150)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
8	email	varchar(254)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
9	is_staff	tinyint(1)			No	None			Change Drop Primary Unique More
10	is_active	tinyint(1)			No	None			Change Drop Primary Unique More
11	date_joined	datetime(6)			No	None			Change Drop Primary Unique More

Figure 4.13 Auth_user

8. auth_permissions

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	name	varchar(255)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	content_type_id	int(11)			No	None			Change Drop Primary Unique More
4	codename	varchar(100)	latin1_swedish_ci		No	None			Change Drop Primary Unique More

Figure 4.14 Auth_permissions

9. auth_Group_permissions

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop Primary Unique Index Spatial More
2	group_id	int(11)			No	None			Change Drop Primary Unique Index Spatial More
3	permission_id	int(11)			No	None			Change Drop Primary Unique Index Spatial More

Figure 4.15 Auth_Group_permissions

10. auth_group

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
	1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop Primary Unique Index More
	2	name	varchar(80)	latin1_swedish_ci		No	*None*			Change Drop Primary Unique Index More

Figure 4.16 Auth_group

4.2 Project Requirement

1. Server To Put Project File in it
2. Browser to access website

4.3 Project Values

1. Can Write Articles in it
2. Can (Comment / Like / Dislike / Delete / Edit) Articles
3. Save time and effort for searching for specific opinion (positive or negative) for specific subject so user don't need to read all articles.

4.4 Special Issues Or Constraints:

1. User data is not deleted because it is used for machine learning and training data for future.
2. Project must be done in one year
3. Project should be easy to use by users

Chapter 5
Implementation

5.1 User Manual

5.1.1 Sign UP

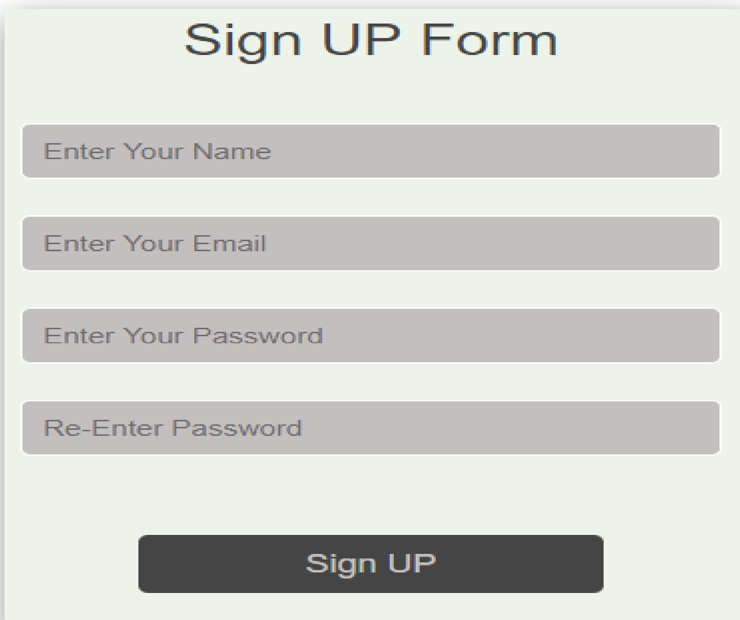


Figure 5.1 Sign UP

Here user can signup To register in website by putting

Name :
should be unique.
should be 40 Characters only

Email :
should be unique.
should be 50 Characters only

Password :
should be 40 Characters only

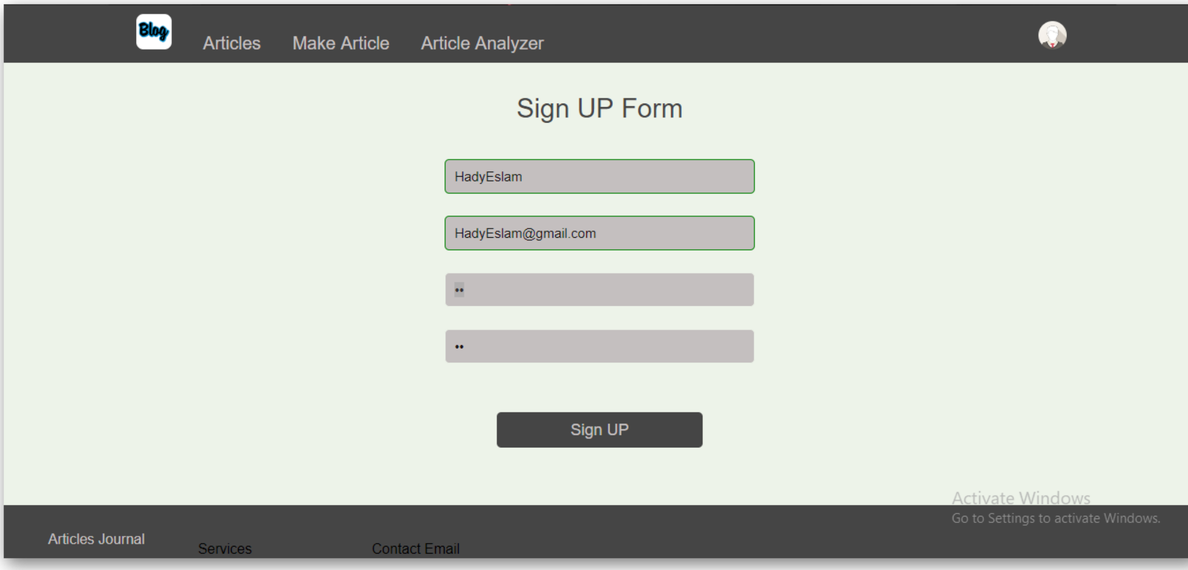
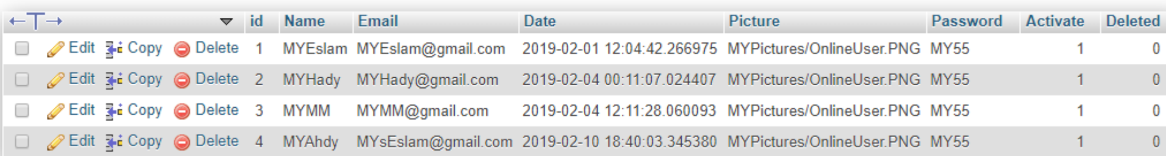


Figure 5.2 Sign UP test



		id	Name	Email	Date	Picture	Password	Activate	Deleted
	Edit Copy Delete	1	MYEslam	MYEslam@gmail.com	2019-02-01 12:04:42.266975	MYPictures/OnlineUser.PNG	MY55	1	0
	Edit Copy Delete	2	MYHady	MYHady@gmail.com	2019-02-04 00:11:07.024407	MYPictures/OnlineUser.PNG	MY55	1	0
	Edit Copy Delete	3	MYMM	MYMM@gmail.com	2019-02-04 12:11:28.060093	MYPictures/OnlineUser.PNG	MY55	1	0
	Edit Copy Delete	4	MYAhdy	MYsEslam@gmail.com	2019-02-10 18:40:03.345380	MYPictures/OnlineUser.PNG	MY55	1	0

Figure 5.3 Sign UP form DB

5.1.2 Log in

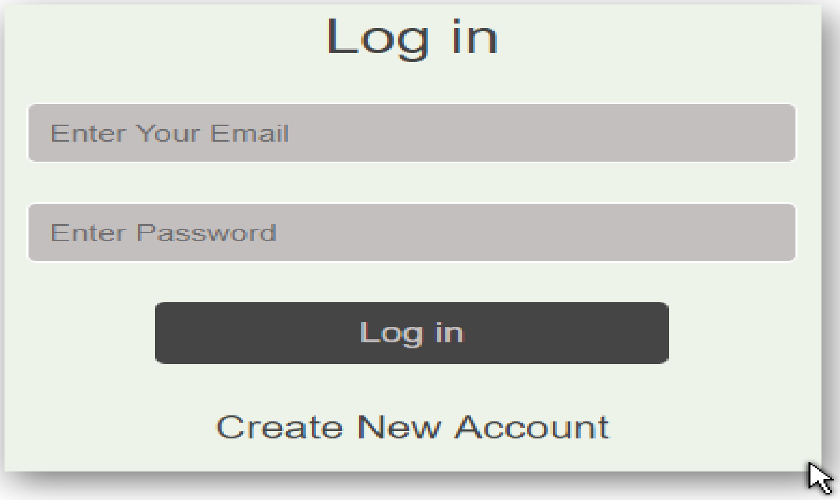


Figure 5.4 Log in form

Here user log in to website by putting:-

Email :
should be unique.
should be 50 Characters only

Password :
should be 40 Characters only

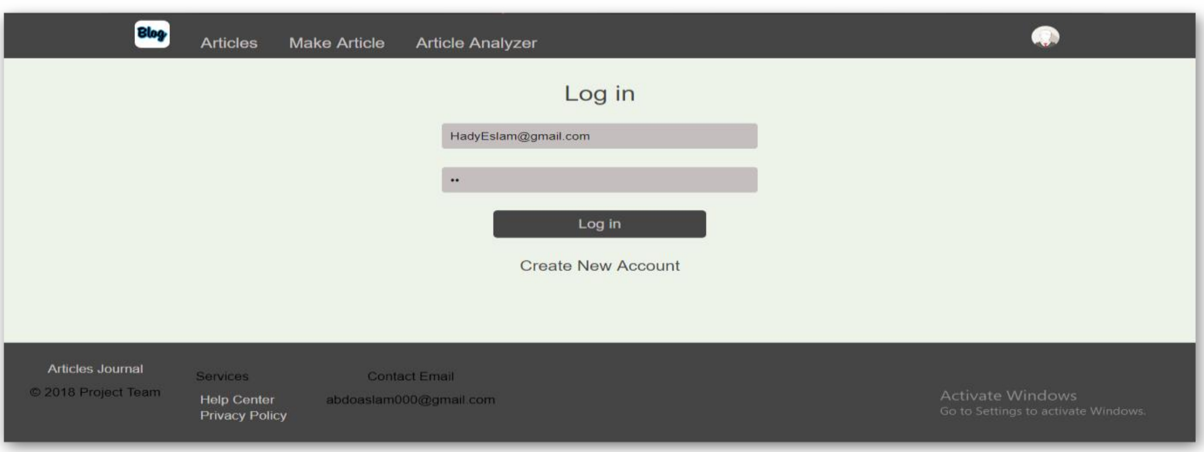


Figure 5.5 Log in test

5.1.3 Log Out “User Can Log Out”

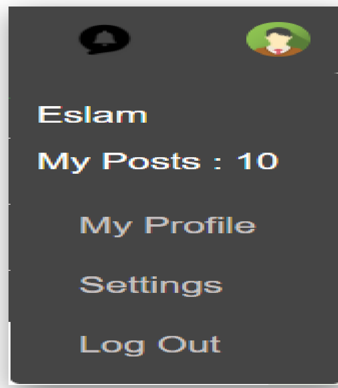


Figure 5.6 Drop down list

5.1.4 Make Articles

Figure 5.7 Make article area

Here registered Users can make Articles by clicking in *make Article* Link

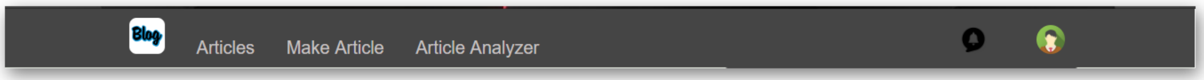


Figure 5.8 Header of page

Do these Steps to post an Article:-

Article Title

should not be more than 100 Characters

Tags

should Be no more than 500 Characters

Article

should Be no more than 2000 Characters

Figure 5.9 Make article area test

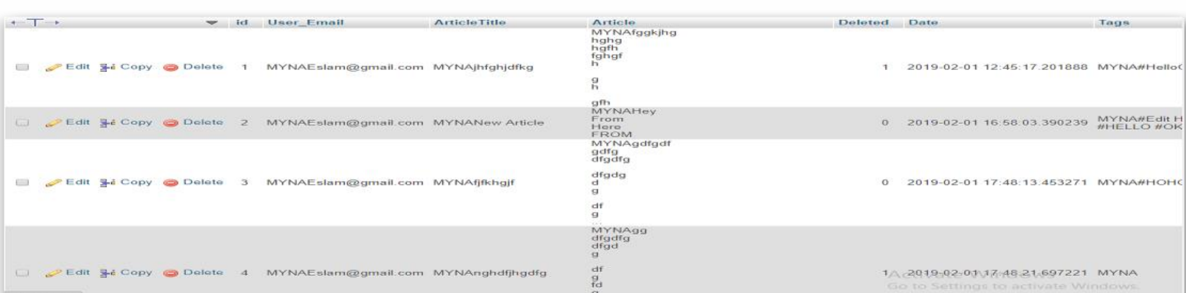


Figure 5.10 Article from DB

5.1.5 Main Page

Here user can see all articles in website and search for articles to see Full Article click on Link [The Link To Full Article]

Figure 5.11 Main Page

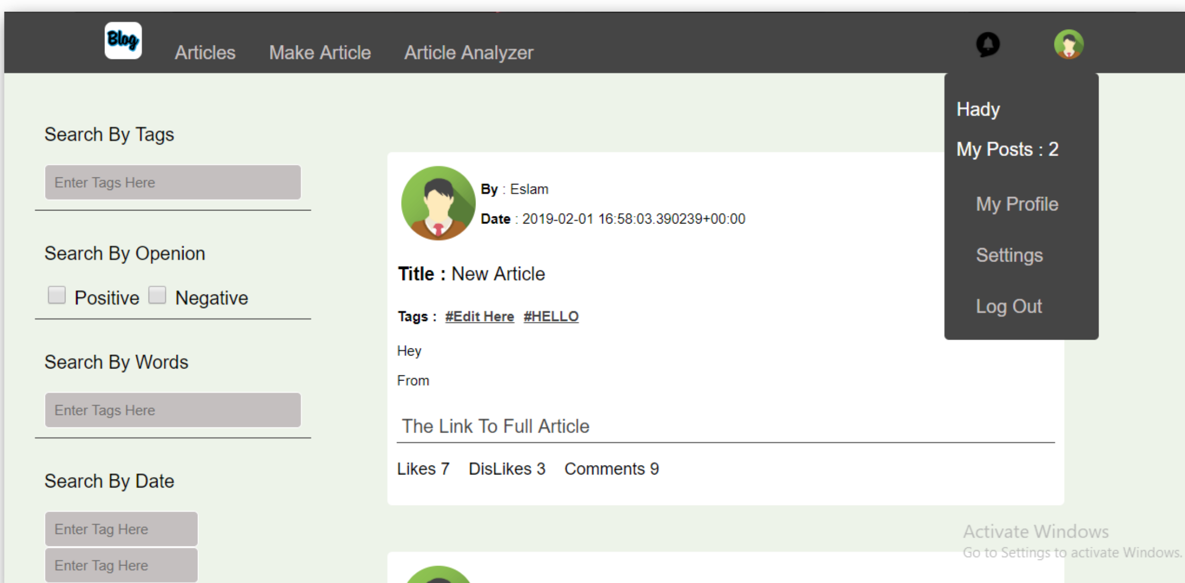


Figure 5.12 Main Page2

5.1.6 My Profile

Here user can see his information's and all of his articles and can do

Edit Articles

By clicking The Arrow in Post The Click Edit Link

Delete Articles

By clicking The Arrow in Post The Click Delete Link

Figure 5.13 My Profile page

5.1.7 User Profile

Here, users can see other user's information's and all his articles

Figure 5.14 User Profile page

5.1.8 Show Article

Here, users can see the whole Article and can do:-

like the Article:
by clicking Like Button
dislike the Article:
by clicking 'Dislike Button'
Comment on the Article:
by writing in Comment text area then clicking in Send Picture Beside it
User can only Like Or Dislike The Article Once And Comment Many Comments if he wants

Figure 5.15 Show article page

Figure 5.16 Comment test

←T→	id	Post_id	User_Email	Comment	Date
Edit Copy Delete	1	1	MYNAMEEslam@gmail.com	MYNAMEdsfdsfdsf	2019-02-02 13:09:42.020483
Edit Copy Delete	2	1	MYNAMEEslam@gmail.com	MYNAMEafdsfdsfsdf	2019-02-02 13:13:00.873235
Edit Copy Delete	3	1	MYNAMEEslam@gmail.com	MYNAMEHello From Here	2019-02-02 13:13:29.384903
Edit Copy Delete	4	1	MYNAMEEslam@gmail.com	MYNAMEHello From Here	2019-02-02 13:14:05.437043
Edit Copy Delete	5	1	MYNAMEEslam@gmail.com	MYNAMEasdas	2019-02-02 13:14:39.378834
Edit Copy Delete	6	1	MYNAMEEslam@gmail.com	MYNAMEdasd	2019-02-02 13:14:53.778061
Edit Copy Delete	7	1	MYNAMEEslam@gmail.com	MYNAMEsdasds	2019-02-02 13:15:16.101803
Edit Copy Delete	8	1	MYNAMEEslam@gmail.com	MYNAMEjkfdhgjfdjhj	2019-02-02 13:16:03.938555
Edit Copy Delete	9	1	MYNAMEEslam@gmail.com	MYNAMEHello	2019-02-02 13:23:46.389031
Edit Copy Delete	10	1	MYNAMEEslam@gmail.com	MYNAMEHello From Here	2019-02-02 13:24:49.509804
Edit Copy Delete	11	1	MYNAMEEslam@gmail.com	MYNAMEHello	2019-02-02 14:35:37.759172
Edit Copy Delete	12	1	MYNAMEEslam@gmail.com	MYNAMEsdasd	2019-02-02 14:39:19.682506
Edit Copy Delete	13	1	MYNAMEEslam@gmail.com	MYNAMEHello From Here	2019-02-02 14:39:49.858290
Edit Copy Delete	14	1	MYNAMEEslam@gmail.com	MYNAMEHI	2019-02-02 14:41:03.463431
Edit Copy Delete	15	1	MYNAMEEslam@gmail.com	MYNAMEsdfgsd	2019-02-02 14:41:21.136907
Edit Copy Delete	16	1	MYNAMEEslam@gmail.com	MYNAMEHAHAHAHAHA	2019-02-02 15:56:52.395295

Figure 5.17 Comment from DB

5.1.9 My Notifications

Here User Can See All Of His Notifications

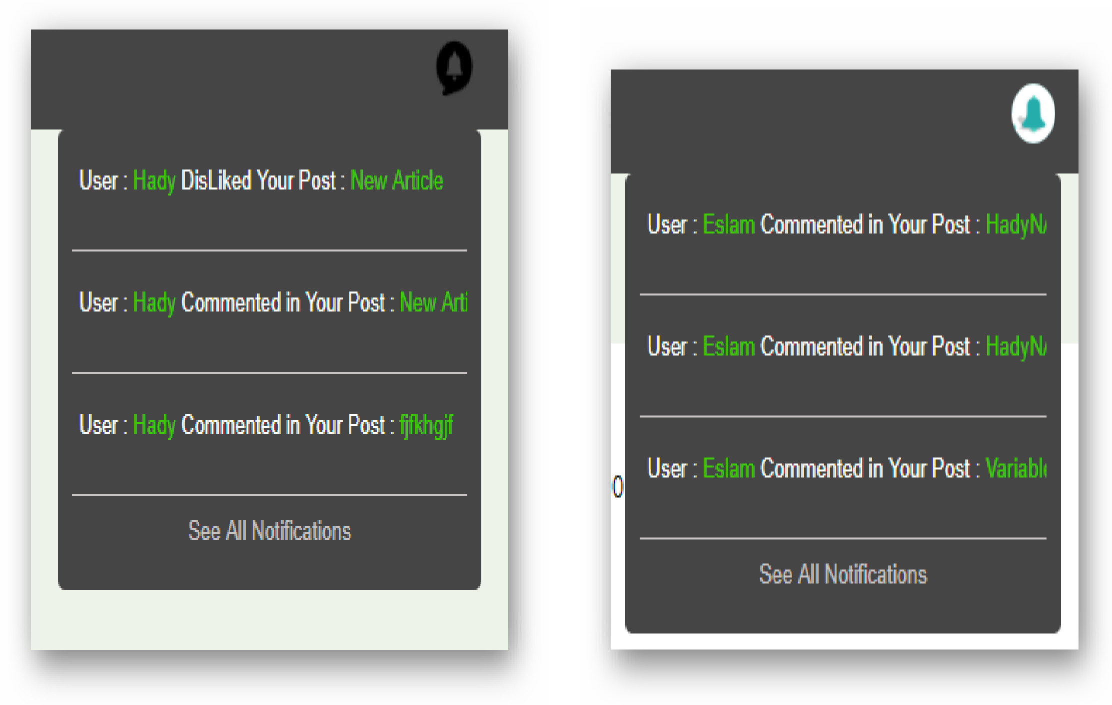


Figure 5.18 My Notifications list

to see All Notifications Click in Link [See All Notifications]

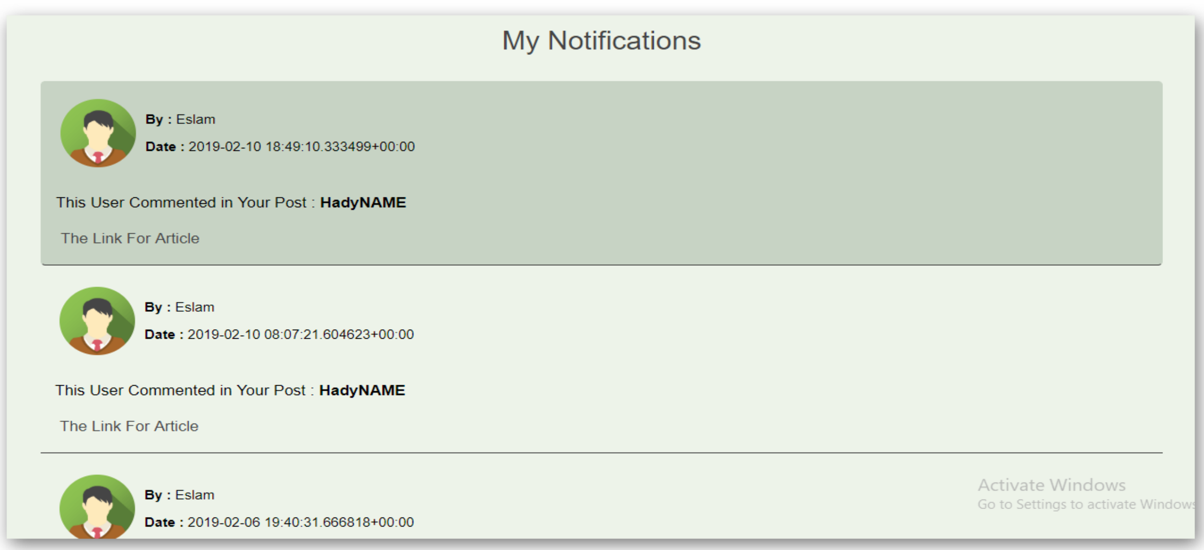
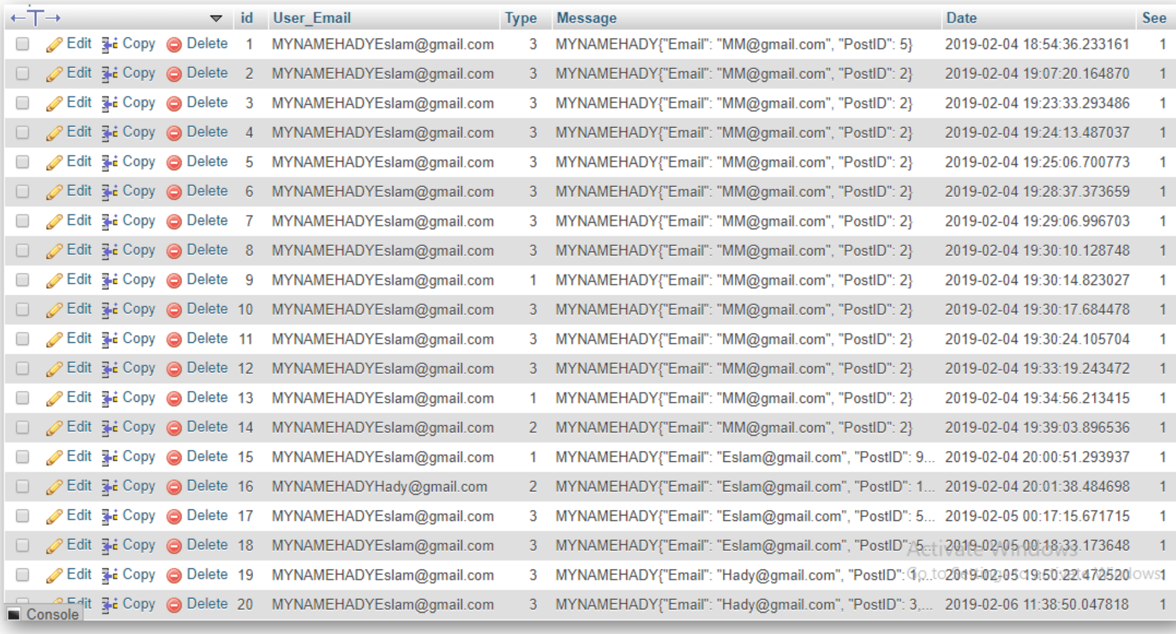


Figure 5.19 My Notifications page



	id	User_Email	Type	Message	Date	See
Edit Copy Delete	1	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 5}	2019-02-04 18:54:36.233161	1
Edit Copy Delete	2	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:07:20.164870	1
Edit Copy Delete	3	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:23:33.293486	1
Edit Copy Delete	4	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:24:13.487037	1
Edit Copy Delete	5	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:25:06.700773	1
Edit Copy Delete	6	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:28:37.373659	1
Edit Copy Delete	7	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:29:06.996703	1
Edit Copy Delete	8	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:30:10.128748	1
Edit Copy Delete	9	MYNAMEHADYEslam@gmail.com	1	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:30:14.823027	1
Edit Copy Delete	10	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:30:17.684478	1
Edit Copy Delete	11	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:30:24.105704	1
Edit Copy Delete	12	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:33:19.243472	1
Edit Copy Delete	13	MYNAMEHADYEslam@gmail.com	1	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:34:56.213415	1
Edit Copy Delete	14	MYNAMEHADYEslam@gmail.com	2	MYNAMEHADY{"Email": "MM@gmail.com", "PostID": 2}	2019-02-04 19:39:03.896536	1
Edit Copy Delete	15	MYNAMEHADYEslam@gmail.com	1	MYNAMEHADY{"Email": "Eslam@gmail.com", "PostID": 9...	2019-02-04 20:00:51.293937	1
Edit Copy Delete	16	MYNAMEHADYHady@gmail.com	2	MYNAMEHADY{"Email": "Eslam@gmail.com", "PostID": 1...	2019-02-04 20:01:38.484698	1
Edit Copy Delete	17	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "Eslam@gmail.com", "PostID": 5...	2019-02-05 00:17:15.671715	1
Edit Copy Delete	18	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "Eslam@gmail.com", "PostID": 5...	2019-02-05 00:18:33.173648	1
Edit Copy Delete	19	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "Hady@gmail.com", "PostID": 1,...	2019-02-05 19:50:22.472520	1
Edit Copy Delete	20	MYNAMEHADYEslam@gmail.com	3	MYNAMEHADY{"Email": "Hady@gmail.com", "PostID": 3,...	2019-02-06 11:38:50.047818	1

Console

Figure 5.20 My Notifications from DB

5.1.10 Settings

Here, user can change his settings like by clicking [Settings] Link from drop down list

He can change:-

1. **Name Setting**

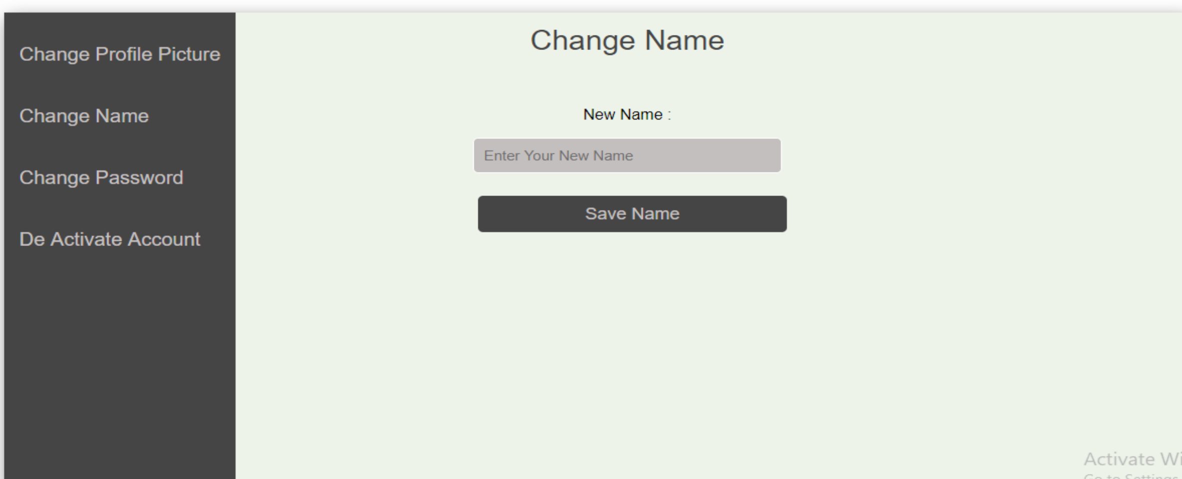


Figure 5.21 Name setting

2. Password Setting

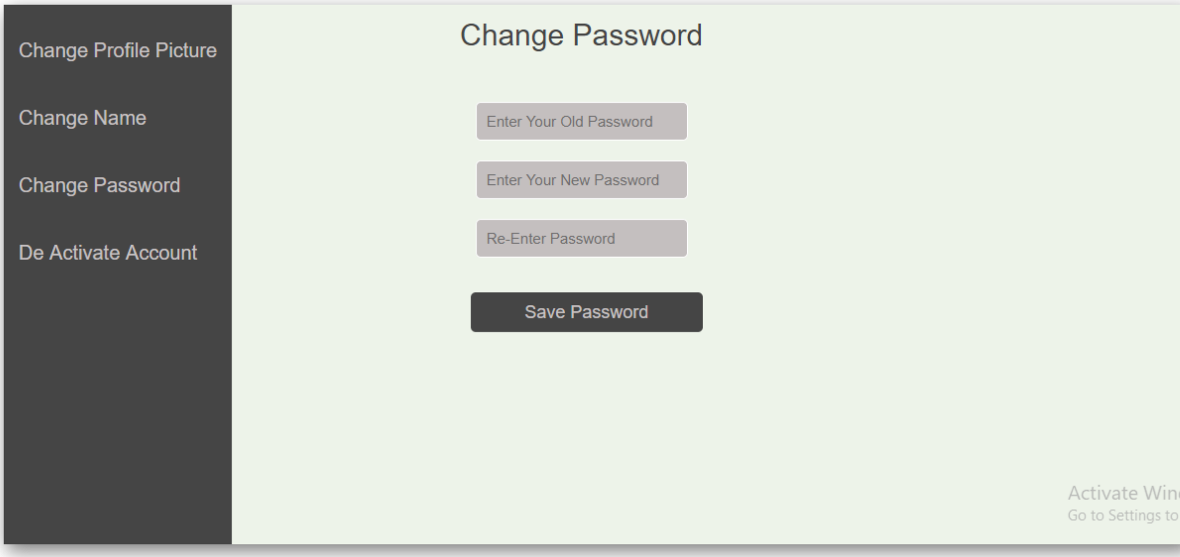


Figure 5.22 Password setting

3. Deactivate Account

By clicking this button, user's profile and articles will not be shown in website

4. Delete Account

By Clicking this button, user's Profile and articles will be deleted from website but it will not be actually deleted so it will be used by website For Machine Learning.

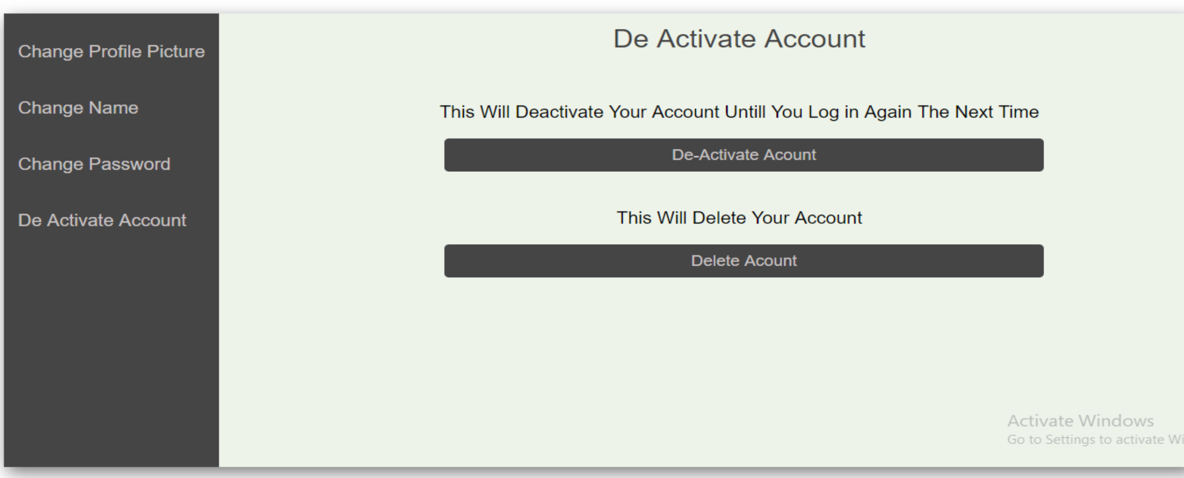


Figure 5.23 Deactivate& Delete

5.1.11 Help Center

Here, user can get help in using Website

5.1.12 Privacy Policy

Here, user can know the policy of the website and the restriction in using website

References

https://www.figure-eight.com/data-for-everyone/

https://www.w3schools.com/html/html_intro.asp

https://www.w3schools.com/css/css_intro.asp

https://www.w3schools.com/js/js_intro.asp

https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

https://www.w3schools.com/jquery/jquery_intro.asp

https://machinelearningmastery.com/clean-text-machine-learning-python/

https://www.geeksforgeeks.org/python-stemming-words-with-nltk/

http://istqbexamcertification.com/what-are-the-software-development-models/