MINISTRY OF EDUCATION وزارة التعليم
UNIVERSITY OF HAFR ALBATIN جامعة حفر الباطن

جامعة حفر الباطن
University of Hafr Al Batin

University Of Hafr Albatin.

Computer Engineering

CSE381

Date: 26/5/2022

Arabic sentiment Analysis

Submit by:

Atheer Alhrabi 219100390.

Abrar Alrashidi 2191003368

Amjad Almutairi 2171003385.

Hanan aljuhani 2191003341

Supervisor: Dr. Aminat Ajibola

Sentiment Analysis System

Development and Evaluation

1. Train Set vs Test Set

This section describes how we divide the produced BoW into Train and Test sets.

Once the BoW representations of the whole dataset produced, the dataset is divided into to sets, Train set and Test Set.

The Train Set contains 900 samples of each class (2700 samples in total) and it is utilized in training the classifiers. The Test Set contains 100 samples of each class (300 samples in to total) and it is utilized in evaluating the trained classifiers.

Each sample in Train Set and Test set must have the correct class label encoded as number. In this work, the codes { -1 , 0 , 1 } are used to represent { negative, neutral , positive } classes, respectively.

The following Python code is used to split the BoW representation into Train/Test Sets and to provide encoded labels for all samples each set:

```
################################ Classification ################################

#1. divide the samples into Train set and Test Set

#1.1. add the first 900 samples of each class in train set
train_set = bow[0:900]
train_set = append(train_set , bow[1000:1900] ,axis=0)
train_set = append(train_set , bow[2000:2900] ,axis=0)

#1.2 add the last 100 samples of each class to test set
test_set = bow[900:1000]
test_set = append(test_set , bow[1900:2000] , axis=0)
test_set = append(test_set , bow[2900:3000] , axis=0)

#1.3. define the class of each sample in the train set as a number: 1=pos, -1=neg , 0=nue
train_classes = array([1 for _ in range(900)] + [-1 for _ in range(900)] + [0 for _ in range(900)])

#1.4. define the class of each sample in the test set as a number: 1=pos, -1=neg , 0=nue
test_classes = array([1 for _ in range(100)] + [-1 for _ in range(100)] + [0 for _ in range(100)])
```

The block of code which is written above is also present in our program as shown below, i-e,

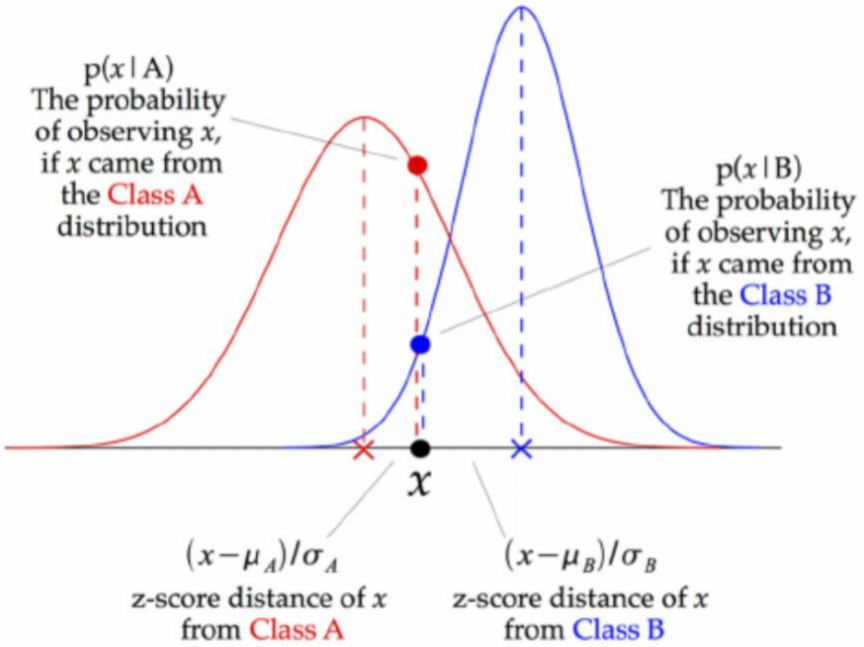
```
##################################### Classification ###################
#1. divide the samples into Train set and Test Set
#1.1. add the first 900 samples of each class in train set
train_set = bow[0:900]
train_set = append(train_set , bow[1000:1900] ,axis=0)
train_set = append(train_set , bow[2000:2900] ,axis=0)
#1.2 add the last 100 samples of each class to test set
test_set = bow[900:1000]
test_set = append(test_set , bow[1900:2000] , axis=0)
test_set = append(test_set , bow[2900:3000] , axis=0)

#1.3. define the class of each sample in the train set as a number: 1=pos, -1=neg , 0=nue
train_classes = array([1 for _ in range(900)] + [-1 for _ in range(900)] + [0 for _ in range(900)])

#1.4. define the class of each sample in the test set as a number: 1=pos, -1=neg , 0=nue
test_classes = array([1 for _ in range(100)] + [-1 for _ in range(100)] + [0 for _ in range(100)])
```

2. Classifiers

This Section gives general introduction about the classifiers we utilized in the project:

1. Naïve Bayes with Gaussian Distribution Gaussian NB Classifier

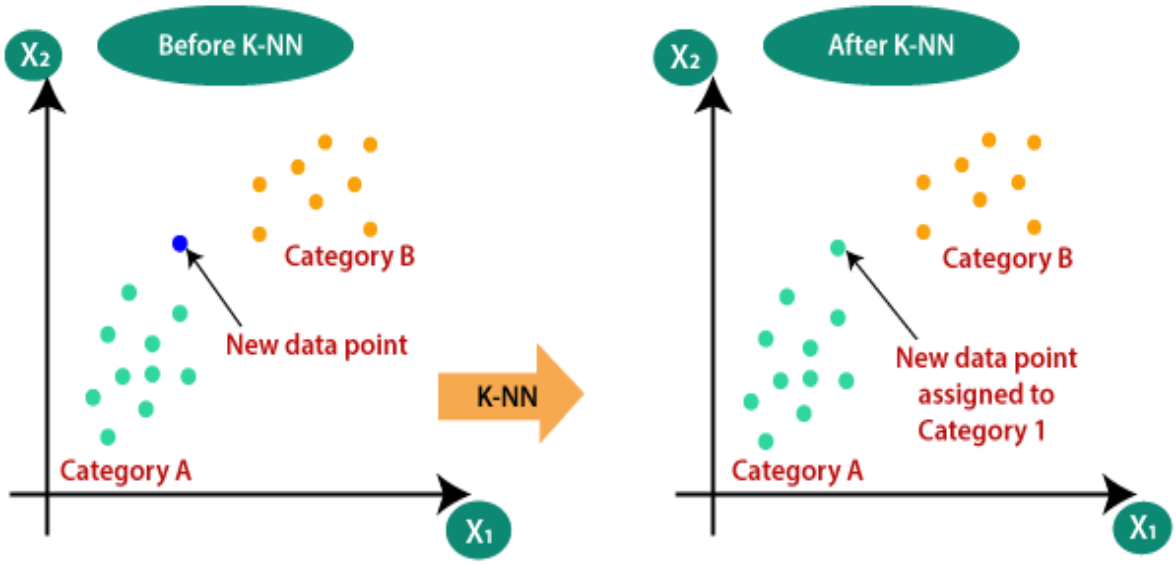
Naïve Bayes is one of the supervised machine learning algorithms which applies bayes theorem principle to calculate the conditional probability of an event based on an assumption that it is equal to the second event's probability. Gaussian Naïve Bayes is a further iteration of Naïve bayes working on continuous features with gaussian distribution, it takes two assumptions while computing, i-e, all features are independent and equal for calculations and computation. Example of Naïve Bayes with Gaussian Distribution is shown below, i-e, [1] [2]



Gaussian Naïve Bayes Working

2. K-Nearest Neighbor KNN Classifier

The biggest issue in this supervised machine learning algorithm is to decide an optimal value of K neighbors. General practice is to keep this value equal to the square root of the value of dataset size under consideration. It is non-parametric and lazy algorithm which means that model distribution is not pre-emptive rather it based on dataset nature. Secondly lazy algorithm implies that it doesn't need training dataset for model creation. After assigning proper value of K neighbors, a test point is put to check the validity of the classifier model. The KNN classifier calculates distance of this test point with respect to all the neighbors and assign a relevant class to this test point as per the acquired results. The working of KNN model is shown in the below figure. [3] [4]

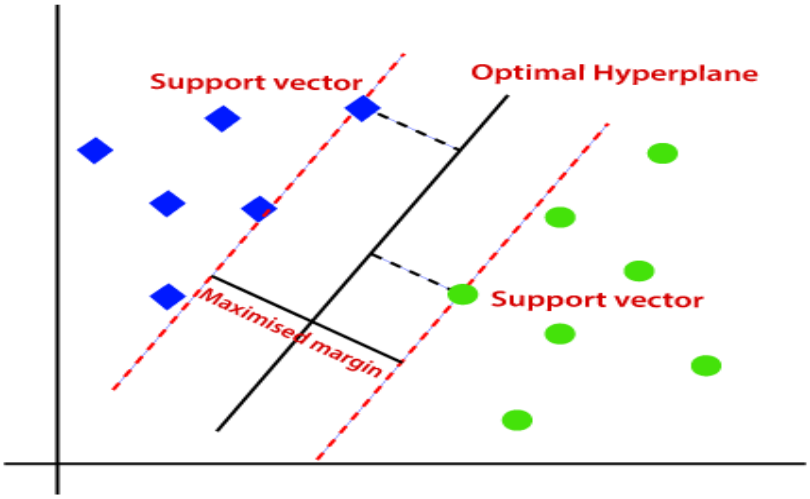


KNN Algorithm Working

3. Support Vector Machine SVM Classifier

This algorithm works to define a proper decision boundary in case of 2 dimensions or a decision plane in case of multi-dimensional features in a dataset. This decision boundary or plane is known as the Hyper Plane. By doing so, this algorithm tends to develop a maximum marginal hyper plane (MMH) to classify the test point into given classes.

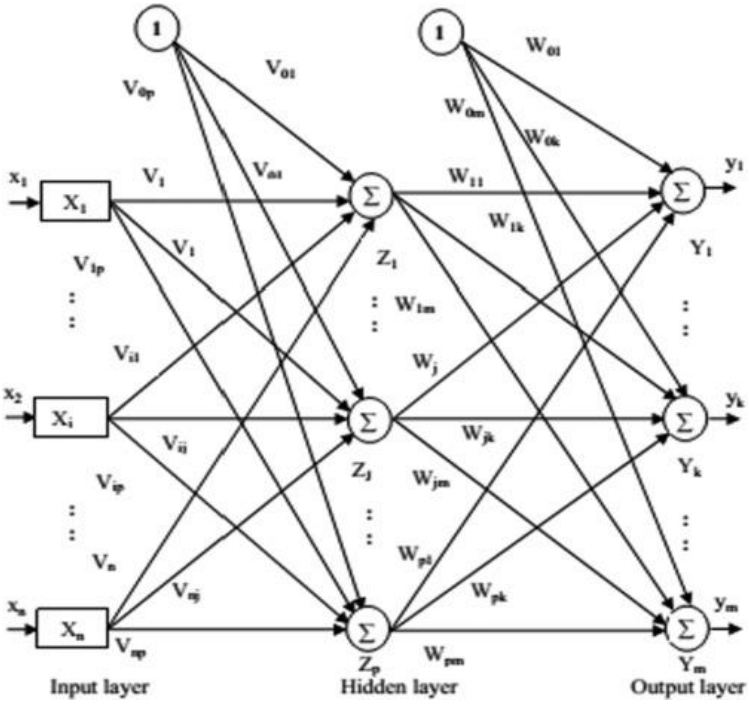
Another main advantage of this algorithm is that it can also used for regression as well as to detect any anomaly in a given dataset. This anomaly detection behavior is known as the outlier detection. The data points nearest to the hyper plane are known as the Support Vector points of marginal hyper plane. This model works with both linear and non-linear dataset models. [5] [6] [7]



SVM Classifier Operation

4. Simple Multi-layer perceptron MLP classifier

MLP classifier is another example of supervised machine learning algorithm which is based on Artificial Neural Networks (ANN) theory. MLP is a feed forward type of ANN which is based on the concept of how a human neuron network works with sensory, associatory and receptor functions. MLP has one input and one output layers and in between them are hidden layers. Number of hidden layers depend on the complexity of work at hand. MLP works on the principle of backpropagation. The initial input wights and bias values are propagated froward for activation and the result values are propagated backwards to adjust the weights and bias values at the input. The working of MLP is shown below, i-e, [8] [2] [4]



MLP classifier Operation

Give comments about the unified 5 steps in the Python of how to train, test and evaluate classifiers.

5 basic steps in python to model, test, train and evaluate any type of classifier are:

1. Creating a classifier in python
2. Training a classifier using training data
3. Testing a classifier using test data
4. Finding number of miscalculated objects
5. Calculating the accuracy of classifier

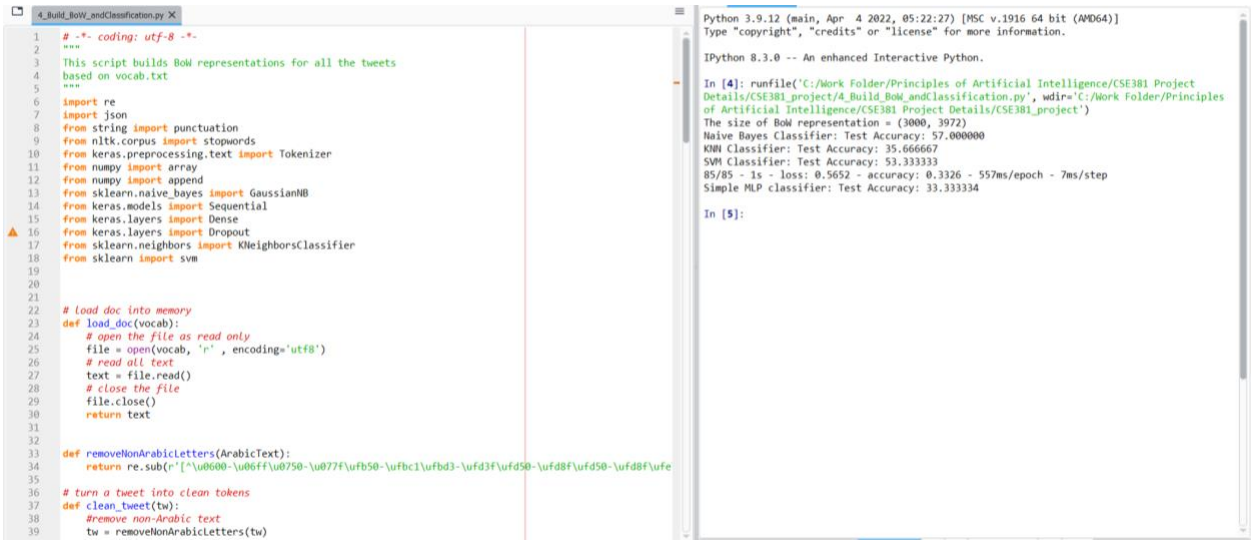
In our code, we have implemented all of these steps while defining any of the above-mentioned classifiers, we can see an example of SVM classifier as follow:

```
153
154     #4. classification using Support Vector Machine classifier
155
156     #Create a svm Classifier
157     svm_linear = svm.SVC(kernel='linear') # Linear Kernel
158
159     #Train the model using the training sets
160     svm_linear.fit(train_set, train_classes)
161
162     #Predict the response for test dataset
163     test_pred = svm_linear.predict(test_set)
164
165     #Counting Number of Miscalculated Objects
166     missing_svm = (test_classes != test_pred).sum()
167
168     #Compute the classification accuracy
169     accuracy= (1 - missing_svm/len(test_classes))
170     print('SVM Classifier: Test Accuracy: %f' % (accuracy*100))
```

3. Evaluation

Run the code and report the Classification Accuracies of the classifiers.

After execution of code from python file “4_Build_BoW_andClassification.py”. We get the accuracies for all the classifiers mentioned above. The accuracy values for all classifiers are shown below along with the running of code, i-e,




```
The size of BoW representation = (3000, 3972)
Naive Bayes Classifier: Test Accuracy: 57.000000
KNN Classifier: Test Accuracy: 35.666667
SVM Classifier: Test Accuracy: 53.333333
85/85 - 1s - loss: 0.5652 - accuracy: 0.3326 - 557ms/epoch - 7ms/step
Simple MLP classifier: Test Accuracy: 33.333334

In [5]:
```

Provide comments about the obtained accuracies.

As per the results, Naïve Bayes classifier is giving us the best accuracy value with 57% of accurate data classification out of all the other classifiers.

References

[1] "Guassian Naive Bayes," [Online]. Available: https://iq.opengenus.org/gaussian-naive-bayes/. [Accessed 23 May 2022].

[2] MonkeyLearn Blog, "https://monkeylearn.com," Monkey Learn, [Online]. Available: https://monkeylearn.com/blog/sentiment-analysis-examples/. [Accessed 08 May 2022].

[3] B. Saji, "a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python," 20 January 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/01/. [Accessed 23 May 2022].

[4] GeeksForGeeks, "https://www.geeksforgeeks.org," 17 July 2020. [Online]. Available: https://www.geeksforgeeks.org/what-is-sentiment-analysis/. [Accessed 08 May 2022].

[5] C. Maklin, "Support Vector Machines," 13 August 2019. [Online]. Available: https://towardsdatascience.com/support-vector-machine-python-example-d67d9b63f1c8. [Accessed 23 May 2022].

[6] A. Naviani, "Support Vector Machines," 19 December 2019. [Online]. Available: https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python. [Accessed 23 May 2022].

[7] "Support Vector Machines," [Online]. Available: https://scikit-learn.org/stable/modules/svm.html. [Accessed 23 May 2022].

[8] C. Bento, "Multi-Layer Perceptron," 21 September 2021. [Online]. Available: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141. [Accessed 23 May 2022].