ASSIGNMENT 1 - SMAI
Ayush Deva
201501098

PROBLEM 1

Observation : For the batch perceptron , it was taking 2184 epochs to converge. To remain in the time limit, I iterate only to 500 epochs and achieve a good accuracy. For the single sample perceptron case, where it converges in 13 and 21 epochs respectively, we observe a better performance with margin as compared to without.

Reasoning : The better performance of the margin algorithm can be attributed to a better separation plane that is in a way equidistant from both the classes, thus ensuring better classification of newer samples.

1. Single Sample Perceptron without Margin
    a. Accuracy : 99.802
2. Single Sample Perceptron with Margin (=1000000)
    a. Accuracy : 99.921
3. Batch Perceptron without Margin
    a. Accuracy : 99.802
4. Batch Perceptron with Margin (=1000000)
    a. Accuracy : 99.802

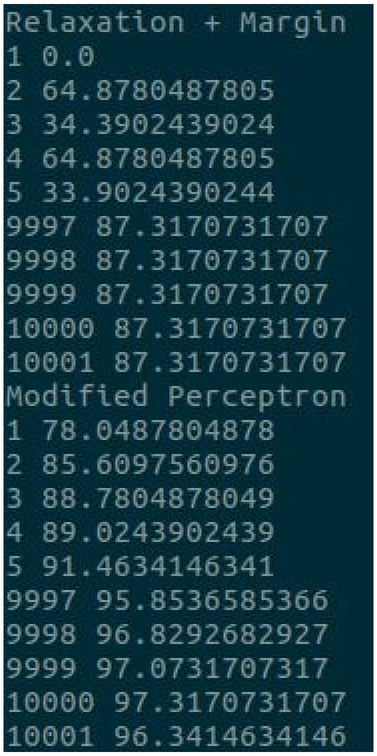
PROBLEM 2

1. Relaxation Algorithm + Margin
    a. Implementation : Here, I have used a version of gradient descent to come up with the most suitable parameter vector for the data. The margin was set to 10. The vector was updated (in a batch fashion) for all the vectors in a epoch where dot(a,x) < margin contributing (margin-dot(a,x))*x / ||x|| to the update.
    b. Accuracy for different epochs * :
        i. First 5 : 0 , 64.87 , 34.39 , 64.87 , 33.9
        ii. Last 5 : 87.31 , 87.31 , 87.31 , 87.31 , 87.31
    c. Accuracy on Validation Set : 99.03 %

    d. Observations : The algorithm is fast , i.e., update in each epoch is faster, but we can probably never find the point of minima. Here, I have exited the loop after 10000 epochs and achieved a good accuracy.

2. Modified Perceptron
    a. Implementation : This is a modified version of the single sample perceptron algorithm. We introduce another hyper parameter which controls how much does a feature vector contribute to the updation of the parameter vector upon misclassification. For this question , we have set it to be (1-accuracy) since better the accuracy on the dataset uptil now, lesser should a misclassified sample should change the parameter vector.
    b. Accuracy for different epochs* :
        i. First 5 : 76.1 , 86.5 , 90 , 89.75 , 91.5
        ii. Last 5 : 96.8 , 97.1 , 97.3 , 96.3 , 96.6
    c. Accuracy on Validation Set : 99.03 %
    d. Observations : The algorithm is fast , i.e., update in each epoch is faster, but we can probably never find the point of minima. Here, I have exited the loop after 10000 epochs and achieved a good accuracy.

*The number of epochs that I ran was 10000, but for the scope for this report have mentioned the first 5 and the last 5 epochs only.



```
Relaxation + Margin
1 0.0
2 64.8780487805
3 34.3902439024
4 64.8780487805
5 33.9024390244
9997 87.3170731707
9998 87.3170731707
9999 87.3170731707
10000 87.3170731707
10001 87.3170731707
Modified Perceptron
1 78.0487804878
2 85.6097560976
3 88.7804878049
4 89.0243902439
5 91.4634146341
9997 95.8536585366
9998 96.8292682927
9999 97.0731707317
10000 97.3170731707
10001 96.3414634146
```

PROBLEM 3

1. Implementation : I have used a binary decision tree to classify the data.

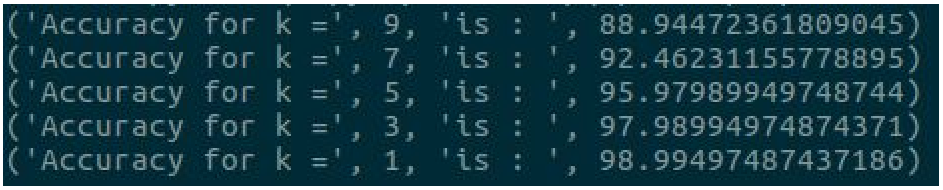
    a. For the real values attributes , I have assigned them to certain ranges (around 10 different range values) depending on its max and min value.
    b. At every decision node, I don’t classify it into one of the value of attributes, instead, every decision node is associated with a attribute and value, and I divide the data as value(+ve) or ~value(-ve).
    c. To choose the decision attribute and value, I have ran a quality check on each possible decision point and chosen the one with the least entropy division.
    d. This is done recursively for every decision node with the dataset it receives from its parent.
    e. My stop condition is when the length of the dataset at that node is less than 50 or the entropy is less than 0.1.

2. Accuracy : 97.3 %

PROBLEM 4

1.
    a. For the BoW representation of the given dataset, I have made a dictionary for each individual document maintaining the count of words that appear in it. This, I felt, would be more efficient that maintaining a count of words in the global vocabulary (present in all documents) as you need not store words that do not appear.
    b. This does not affect the distance calculation function since only the words that appear in both the documents add to the distance. To calculate the distance I am using the **cosine distance** measure .
    c. As part of feature engineering, I am not storing the count of stopwords (articles, determinants etc.) as they appear in every document and do not add relevance to the document.

2.
    a. The accuracies for different values of K are written below. Note that these values can change with the change in validation data. But, we see, that the accuracy for this particular question increases as K decreases.
        i. K = 9 : 88.94
        ii. K = 7 : 92.46
        iii. K = 5 : 95.98
        iv. K = 3 : 97.99
        v. K = 1 : 98.99



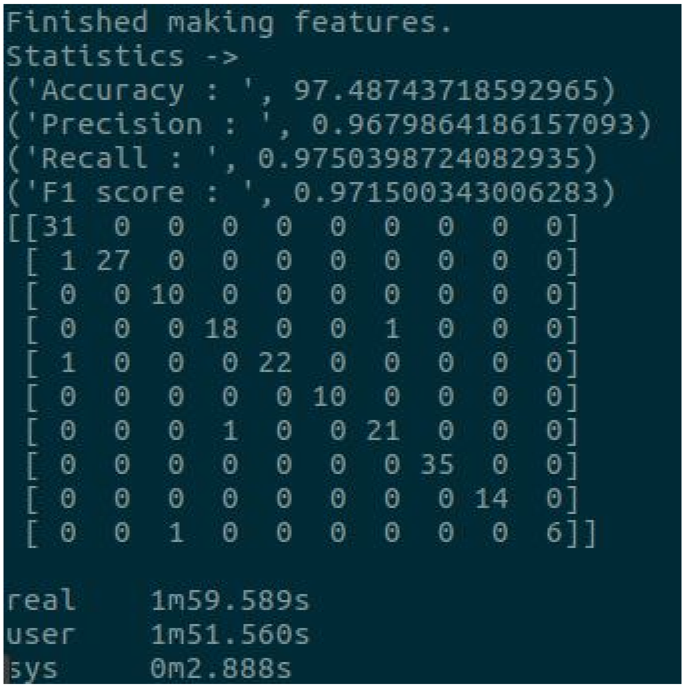
```
('Accuracy for k =', 9, 'is : ', 88.94472361809045)
('Accuracy for k =', 7, 'is : ', 92.4623115577895)
('Accuracy for k =', 5, 'is : ', 95.97989949748744)
('Accuracy for k =', 3, 'is : ', 97.98994974874371)
('Accuracy for k =', 1, 'is : ', 98.99497487437186)
```

    b. Accuracy in the range of 95-99% (due to random.shuffle for choosing validation set).

c. For k=1 , in the last iteration I did before submission, I got the following scores :

   i. Accuracy - 97.48 %
   ii. Precision - 0.968
   iii. Recall - 0.975
   iv. F1 score - 0.971
   v. Confusion Matrix -

```
[31  0  0  0  0  0  0  0  0  0]
[ 1 27  0  0  0  0  0  0  0  0]
[ 0  0 10  0  0  0  0  0  0  0]
[ 0  0  0 18  0  0  1  0  0  0]
[ 1  0  0  0 22  0  0  0  0  0]
[ 0  0  0  0  0 10  0  0  0  0]
[ 0  0  0  1  0  0 21  0  0  0]
[ 0  0  0  0  0  0  0 35  0  0]
[ 0  0  0  0  0  0  0  0 14  0]
[ 0  0  1  0  0  0  0  0  0  6]
```



```
Finished making features.
Statistics ->
('Accuracy : ', 97.48743718592965)
('Precision : ', 0.9679864186157093)
('Recall : ', 0.9750398724082935)
('F1 score : ', 0.971500343006283)
[[31  0  0  0  0  0  0  0  0  0]
 [ 1 27  0  0  0  0  0  0  0  0]
 [ 0  0 10  0  0  0  0  0  0  0]
 [ 0  0  0 18  0  0  1  0  0  0]
 [ 1  0  0  0 22  0  0  0  0  0]
 [ 0  0  0  0  0 10  0  0  0  0]
 [ 0  0  0  1  0  0 21  0  0  0]
 [ 0  0  0  0  0  0  0 35  0  0]
 [ 0  0  0  0  0  0  0  0 14  0]
 [ 0  0  1  0  0  0  0  0  0  6]]

real	1m59.589s
user	1m51.560s
sys	0m2.888s
```