Bangladesh University of Engineering and Technology

CSE472: Machine Learning Sessional

Report on Text Classification

Submitted by-
Md Salman Shamil
Student ID: 1505021

October 16, 2020

Contents

1 Output on Validation Set **2**
1.1 k-Nearest Neighbor 2
1.2 Naive Bayes 2

2 Output on Test Set **2**

3 Results and *t*-statistic **4**

1 Output on Validation Set

1.1 k-Nearest Neighbor

	k=1	k=3	k=5
Hamming distance	$41 \cdot 090909$	$41 \cdot 090909$	$40 \cdot 818182$
Euclidean distance	$57 \cdot 227273$	$57 \cdot 045455$	$57 \cdot 045455$
Cosine similarity with TF-IDF	$81 \cdot 045455$	$83 \cdot 272727$	$83 \cdot 954545$

Table 1: Validation accuracy (in percentage) for different combinations of the distance measure and the values of hyperparameter k. The validation accuracy is highest for Cosine similarity with TF-IDF as distance measure with k=5.

1.2 Naive Bayes

α	Accuracy
$1 \cdot 0$	$91 \cdot 454545$
$0 \cdot 5$	$91 \cdot 863636$
$0 \cdot 25$	$92 \cdot 136364$
$0 \cdot 1$	$92 \cdot 0$
$0 \cdot 0625$	$92 \cdot 227273$
$0 \cdot 05$	$92 \cdot 181818$
$0 \cdot 03125$	$92 \cdot 363636$
$0 \cdot 025$	$92 \cdot 227273$
$0 \cdot 0125$	$92 \cdot 136364$
$0 \cdot 00625$	$92 \cdot 090909$

Table 2: Validation accuracy (in percentage) for different values for smoothing factor(α). The validation accuracy is highest for $\alpha = 0 \cdot 03125$

2 Output on Test Set

Hyperparameters used for test set prediction:

- k-NN: k=5, distance measure: Cosine similarity with TF-IDF
- NB: $\alpha = 0 \cdot 03125$

Iteration	k-NN	NB
1	$86 \cdot 36363636363636$	$92 \cdot 72727272727272$
2	$89 \cdot 0909090909091$	$94 \cdot 54545454545455$
3	$84 \cdot 54545454545455$	$96 \cdot 36363636363636$
4	$82 \cdot 72727272727273$	$90 \cdot 0$
5	$86 \cdot 36363636363636$	$94 \cdot 54545454545455$
6	$87 \cdot 27272727272727$	$93 \cdot 63636363636364$
7	$87 \cdot 27272727272727$	$95 \cdot 45454545454545$
8	$87 \cdot 27272727272727$	$91 \cdot 81818181818183$
9	$80 \cdot 9090909090909$	$91 \cdot 81818181818183$
10	$84 \cdot 54545454545455$	$90 \cdot 9090909090909$
11	$83 \cdot 63636363636363$	$93 \cdot 63636363636364$
12	$86 \cdot 36363636363636$	$95 \cdot 45454545454545$
13	$77 \cdot 27272727272727$	$88 \cdot 18181818181819$
14	$81 \cdot 81818181818183$	$89 \cdot 0909090909091$
15	$81 \cdot 81818181818183$	$91 \cdot 81818181818183$
16	$77 \cdot 27272727272727$	$87 \cdot 27272727272727$
17	$79 \cdot 0909090909091$	$90 \cdot 9090909090909$
18	$78 \cdot 18181818181819$	$91 \cdot 81818181818183$
19	$76 \cdot 36363636363637$	$90 \cdot 9090909090909$
20	$78 \cdot 18181818181819$	$88 \cdot 18181818181819$
21	$80 \cdot 0$	$86 \cdot 36363636363636$
22	$88 \cdot 18181818181819$	$94 \cdot 54545454545455$
23	$86 \cdot 36363636363636$	$91 \cdot 81818181818183$
24	$81 \cdot 81818181818183$	$93 \cdot 63636363636364$
25	$84 \cdot 54545454545455$	$89 \cdot 0909090909091$
26	$83 \cdot 63636363636363$	$90 \cdot 0$
27	$81 \cdot 81818181818183$	$87 \cdot 27272727272727$
28	$78 \cdot 18181818181819$	$90 \cdot 9090909090909$
29	$81 \cdot 81818181818183$	$89 \cdot 0909090909091$
30	$80 \cdot 0$	$94 \cdot 54545454545455$
31	$85 \cdot 45454545454545$	$90 \cdot 0$
32	$80 \cdot 9090909090909$	$92 \cdot 72727272727272$
33	$84 \cdot 54545454545455$	$92 \cdot 72727272727272$
34	$82 \cdot 72727272727273$	$94 \cdot 54545454545455$
35	$80 \cdot 9090909090909$	$89 \cdot 0909090909091$
36	$80 \cdot 0$	$90 \cdot 0$
37	$80 \cdot 9090909090909$	$89 \cdot 0909090909091$
38	$78 \cdot 18181818181819$	$91 \cdot 81818181818183$

Iteration	k-NN	NB
39	88 · 18181818181819	97 · 27272727272728
40	87 · 27272727272727	95 · 45454545454545
41	77 · 27272727272727	89 · 0909090909091
42	88 · 18181818181819	96 · 36363636363636
43	81 · 81818181818183	88 · 18181818181819
44	80 · 9090909090909	89 · 0909090909091
45	83 · 63636363636363	92 · 72727272727272
46	84 · 54545454545455	93 · 63636363636364
47	83 · 63636363636363	90 · 9090909090909
48	80 · 0	89 · 0909090909091
49	81 · 81818181818183	89 · 0909090909091
50	85 · 45454545454545	94 · 54545454545455

Table 3: Test accuracy for 50 iterations with k-NN and Naive Bayes. Each iteration contains 10 documents from each topic.

3 Results and *t*-statistic

Accuracy values on test set with 50 iterations showed the following results:
k-NN

- **Mean**: 82 · 78181818181818
- **Standard Deviation**: 3 · 3986385217691044

Naive Bayes

- **Mean**: 91 · 63636363636364
- **Standard Deviation**: 2 · 7029215904215462

t-statistic was calculated using SciPy function scipy.stats.ttest_rel(a, b, axis=0, nan_policy='propagate'). The results for *t*-statistic calculation are as follows:

- ***t*-statistic**: 23 · 64796533269011
- ***p*-value**: 1 · 8721014627256132e−28

Here *p*-value is less than all the values of significance level α (0·005, 0·01 and 0·05). Consequently, *p*-value $\leq \alpha$ implies that we can reject the null hypothesis that the means are equal. It suggests that the output difference between the two algorithms is statistically significant and the higher accuracy of Naive Bayes is not because of

some random fluke.
So from the mean test accuracy and t-statistic, it can be decided that Naive Bayes algorithm performed better than k-NN.