Assignment 2

1st Abdulkarim Dawalibi
2nd Adnan Altukleh

I. Introduction

The number of spam emails received in our inboxes has slightly increased in the last decade [1]. For this reason, we are going to train a model to predict if an email is a spam or not.

We are going to use three different supervised algorithms which are: Naive Bayes, Random Forest and KNN. Cross validation method will be used on the data to train and test the models. The dataset will be used to solve the problem called 'Spambase Data Set' which consists of 57 attributes and 4601 instances [2].

II. Method

A. Data cleaning

Started by checking for missing values and detecting outliers. Also dropped all existing duplicated instances.

B. Preprocessing

In this step, we have created a copy of our dataset. On the first data frame we class discretized the data using binning with help of the Sklearn KBinsDiscretizer library. We used the kmeans strategy, decided to make 8 bins and encode the data ordinal. We fitted the data and then transformed the data [4]. On the second data frame didn't apply any class discretization.

C. Cross validation

The method was implemented by using Sklearn library. We use Stratified k-fold class with 10 as number of splits. Then we split each fold to a train and test data.

D. Implementing the algorithms

We implemented the algorithms with the help of Sklearn library.

E. Train the models

By applying cross validation, the data were divided into ten blocks. In each one of the ten folds, each fold contained 90% train data and 10% as test data. In each fold the train blocks and test block change, i.e., blocks number 1-9 as train data and block number 10 as test data, in the next iteration blocks number 1-8 and 10 as train data and block number 9 as test data, etc.
Using cross validation method, we trained each model ten times with data discretization and ten time without data discretization.

F. Evaluate the models

To evaluate the performance of the models we measured the time it took for each model to train, calculated the accuracy of the trained models and computed the F-measure. All measurements were added to separate tables for comparison. The layout of the tables is like the table example 12.4 in the course book [3]. We have also computed the average value and standard deviation for each model in the tables. The measurements were also visualized to simplify the comparison between the models with discretized and non-discretized train and test data.

G. Freidman test

We implemented the statistical test from scratch. First, we ranked the performance of each model on all our tables (accuracy, f-measure, and time), we computed the average rank for each model ranks. We defined our hypothesis.

H0: No significant difference exists between the avg ranks on the alpha level of 0.05.
H1: At least one significant difference exists between the avg ranks on the alpha level of 0.05.

and tested if any significant difference exists between the models using the following formula from the course book [4].

$$n\sum_j\left(R_j-\bar{R}\right)^2 \qquad (1)$$

$$\frac{1}{n(k-1)}\sum_{ij}\left(R_{ij}-\bar{R}\right)^2 \quad (2)$$

H. Nemenyi test

We calculated the critical difference with the help of the formula in the course book(3), to find out which models have a significant difference between each other [5].

$$CD = q_a\sqrt{\frac{k(k+1)}{6n}} \quad (3)$$

III. Results

As a result of data cleaning, we got 391 duplicated instances otherwise, no outliers nor missing values were detected.

Evaluating the models gave the following results, in the accuracy measurement the random forest model with non-discretized train and test data had an average accuracy result around 95%.

	Fold	Naive Bayes	Naive Bayes D	Random Forest	Random Forest D	KNN	KNN D
0	1	0.798100	0.904988	0.950119	0.926366	0.750594	0.876485
1	2	0.790974	0.921615	0.952494	0.940618	0.764846	0.909739
2	3	0.800475	0.919240	0.923990	0.938242	0.764846	0.912114
3	4	0.821853	0.928741	0.954869	0.935867	0.807601	0.912114
4	5	0.828979	0.916865	0.957245	0.942993	0.802850	0.912114
5	6	0.800475	0.895487	0.952494	0.919240	0.800475	0.916865
6	7	0.819477	0.928741	0.961995	0.957245	0.798100	0.914489
7	8	0.821853	0.907363	0.966746	0.954869	0.826603	0.912114
8	9	0.660333	0.662708	0.893112	0.840855	0.710214	0.831354
9	10	0.762470	0.750594	0.850356	0.878860	0.762470	0.836105
10	avg	0.793613	0.887305	0.945896	0.928477	0.780681	0.899710
11	stdev	0.051720	0.084949	0.023114	0.034999	0.036043	0.028374

Figure 1: Accuracy

	Fold	Naive Bayes	Naive Bayes D	Random Forest	Random Forest D	KNN	KNN D
0	1	0.736842	0.888889	0.935780	0.905199	0.680851	0.832258
1	2	0.717949	0.907563	0.939394	0.924012	0.702703	0.880503
2	3	0.734177	0.904494	0.898734	0.920732	0.691589	0.883281
3	4	0.766355	0.916201	0.941538	0.916923	0.742857	0.884735
4	5	0.770701	0.900285	0.945783	0.928144	0.756598	0.884013
5	6	0.742331	0.879121	0.941176	0.901163	0.751479	0.898551
6	7	0.756410	0.914773	0.951220	0.944785	0.728435	0.885350
7	8	0.758842	0.888252	0.957576	0.941896	0.772586	0.883281
8	9	0.632391	0.683036	0.873950	0.818428	0.668478	0.782875
9	10	0.696970	0.724409	0.809668	0.838095	0.700599	0.775244
10	avg	0.735111	0.875846	0.931683	0.911254	0.721731	0.868316
11	stdev	0.042074	0.073381	0.027233	0.037730	0.037006	0.036941

Figure 1: F-measure

	Fold	Naive Bayes	Naive Bayes D	Random Forest	Random Forest D	KNN	KNN D
0	1	0.059540	0.002994	0.742360	0.839387	0.003999	0.001996
1	2	0.009005	0.004003	0.831838	0.483515	0.002998	0.000998
2	3	0.006000	0.007012	0.790726	0.662301	0.003978	0.001046
3	4	0.006000	0.004004	0.733453	0.556149	0.003000	0.001000
4	5	0.004998	0.003999	0.964626	0.546344	0.006005	0.002008
5	6	0.006000	0.003004	0.982256	0.632964	0.005056	0.001019
6	7	0.004964	0.002000	0.774305	1.165510	0.003999	0.007003
7	8	0.008006	0.003001	0.757553	0.717999	0.003986	0.002000
8	9	0.004004	0.003003	0.807222	0.709001	0.001999	0.001006
9	10	0.008001	0.002001	0.876301	0.508017	0.001997	0.001997
10	avg	0.012058	0.003669	0.820482	0.701463	0.003891	0.002009
11	stdev	0.017872	0.001418	0.092133	0.204205	0.001175	0.001933

Figure 3: Training-Time

The Friedman statistic results in accuracy, time and F-measure was as follows (15.5, 18.2, 15.5), and the critical value was 7.8. The null hypothesis was rejected.

The results from the Nemenyi test are presented in figure 4.

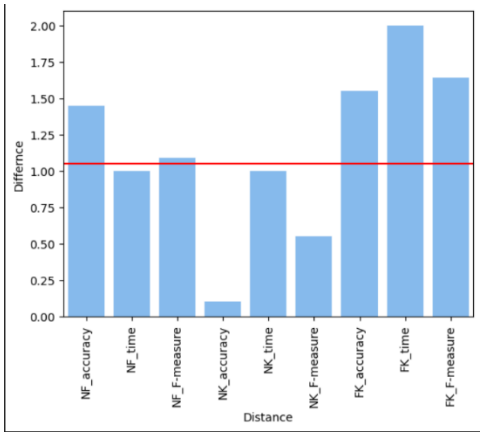

Figure 4: Results from the Nemenyi

NF represent Naive Bayes and Random Forest, NK represent Naive Bayes and KNN, FK represent Random Forest and KNN

IV. Discussion

We dropped the duplicate instances from the dataset to avoid having the same instance in the train data, in this way we guarantee that our model doesn't train on the same instance more than once in this way we avoid overfitting.

In the preprocessing step we decided to make a copy of our dataset were on the first one we didn't apply discretization method on the train and test data. On the second dataset we did. We used a kmeans strategy and 8 bins because the values in each bin have the same nearest center of a 1D k-means cluster. If we tune the parameters such as bin-size and the strategy, we get lower accuracy. Because of the higher number of common features. We made two datasets to observe if the models preform better with or without discretization.

We noticed that overall, all the algorithms preformed better with the discretization, except the random forest model which gave slightly less average accuracy and F-measure results. The accuracy results with-without discretization (95% - 93%). On the other hand, the time for training the models was faster with discretization (0.43s) without discretization the time was (0.59s). This different in time is because the models trained on non-discretized data had to deal with continuous values. We preferred the result we got from the accuracy and F-measure over time result therefore we decided to continue with non-discretized random forest models. And for the other models, we decided to continue working with the discretized trained and tested models.

The Friedman test showed a significant difference between the models this caused by the different behavior of the models. Random forest was ranked as the best performed algorithm due to the benefit of ensemble learning. Where the algorithm creates multiple decision trees, and the output of the algorithm is a class selected by most trees.

Because we had a significant difference from the Friedman test, we wanted to determine which algorithms had a significant difference. Therefor we used Nemenyi test to determine where the significance difference is located.

The Nemenyi test showed that there is a significant difference between Naive Bayes and Random Forest and significant difference between Random Forest and KNN this means that the difference between the classifiers is unlikely to be due to chance and is instead likely to be a true difference in their performance. In the case of where we had a significant difference between Random Forest and KNN a possible implication could indicate that one classifier is more robust or generalizable to different situations than the other.

V. Conclusion

The performance of the chosen algorithms in some of the algorithms was better with discretized data.

The Friedman test and Nemenyi test are statistical methods that are used in conjunction to evaluate the performance of multiple classifiers or algorithms. The Friedman test is performed first to determine if there is a significant difference in the performance of the classifiers overall. If the Friedman test shows that there is a significant difference, the Nemenyi test is then used to identify which pairs of classifiers have significantly different performance from one another. These tests allow us to compare the effectiveness of different models and determine which ones are the most effective for a specific task.

Overall, obtaining a significant difference between classifiers using the Nemenyi test and critical difference can indicate that one classifier is consistently outperforming the other, is more robust, or requires further investigation. Which might be an interesting topic to investigate more in.

REFERENCES

[1] Moorthy, J. (2022). "23 Email Spam Statistics to Know in 2022". mailmodo. https://www.mailmodo.com/guides/email-spam-statistics/

[2] Hopkins, M., Reeber, E., Forman, G., & Suermondt, J. (1999). SpambaseDataSet.UCI. https://archive.ics.uci.edu/ml/datasets/Spambase

[3] FLACH, P. (2012). MACHINE LEARNING The Art and Science of Algorithms that Make Sense of Data. CAMBRIDGE UNIVERSITY PRESS. Pp. 424- 425.

[4] FLACH, P. (2012). MACHINE LEARNING The Art and Science of Algorithms that Make Sense of Data. CAMBRIDGE UNIVERSITY PRESS. Pp. 430- 431.

[5] FLACH, P. (2012). MACHINE LEARNING The Art and Science of Algorithms that Make Sense of Data. CAMBRIDGE UNIVERSITY PRESS. Pp. 431.