Compare the computational and predictive performance of learning algorithms on a spam detection task.

Morris Simons
M.Sc Ai and Machine Learning
Blekinge Tekniska Högskola
Karlskrona, Blekinge, Sweden
Mosi21@student.bth.se

***Abstract*—In this assignment four different supervised classification models are compared using the spambase dataset. The three models that were tested are: K-Nearest neighbors, Random Forest and Support Vector Clustering(SVC). Friedman And Nemenyi Tests were used to test the algoritms**

***Index Terms*—KNN, SVC, Random Forest, Friedman test, Nemenyi test**

I. Introduction

In the pursuit of solving the given task, the objective was initially to employ three distinct supervised classification models. Then later useing the Friedman and Nemenyi test to calcultate the diffrent performance on training time, accuracy and F1 score. To statically prove the significant difference of the selected models on an spam traning dataset.

II. CLASSIFIERS

A. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet powerful algorithm used for classification and regression tasks. It predicts the class or value of a new data point by analyzing its "k" nearest neighbors in the training dataset. KNN operates under the assumption that data points close to each other in feature space are likely to belong to the same class or exhibit similar behavior.

B. Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to enhance prediction accuracy and mitigate overfitting. Each decision tree in the forest is trained on a random subset of the data and features. The final prediction is based on a majority vote or averaging of individual tree predictions. Random Forest is known for its versatility and ability to handle high-dimensional and complex data.

C. Support Vector Clustering (SVC)

Support Vector Clustering (SVC) is primarily used for unsupervised clustering tasks. Unlike KNN, which assigns labels based on neighbor proximity, SVC identifies natural data groupings by maximizing the margin between clusters. It operates in a high-dimensional space and is useful for handling non-linearly separable data. SVC has applications in image segmentation, anomaly detection, and gene expression analysis.

III. FRIEDMAN AND NEMENYI TESTS

The Friedman test is a non-parametric test used to compare means between data sets. The Nemenyi test is used to find the groups of data that differ from each other after the Friedman test has determined that there is a difference.

Friedman's Test Formula

The Friedman test is a non-parametric statistical test used to detect differences in treatments across multiple test attempts. The formula for Friedman's test is as follows:

$$\bar{R} = \frac{1}{nk} \sum_{ij} R_{ij} = \frac{k+1}{2} \tag{1}$$

$$n \sum_{ij} (R_j - \bar{R})^2 \tag{2}$$

$$\frac{1}{n(k-1)} \sum_{ij} (R_{ij} - \bar{R})^2 \tag{3}$$

The formulas provided are part of the theoretical underpinning for the Friedman test but are not directly equivalent to the simplified formula used in our Python implementation used in statistic software such as mintlab. This version looks like this

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]$$

where:

- χ_F^2 is is the Friedman statistic.
- N is the number of subjects
- k is the number of conditions
- R_j is the sum of ranks for the j-th condition

Nemenyi Test Formula

The Nemenyi test serves as a post-hoc analysis, typically following the Friedman test, to conduct pairwise comparisons between multiple groups or treatments. It is particularly useful for assessing the differences in performance of various algorithms or methods across several datasets. When the Friedman test suggests significant overall differences, the Nemenyi test steps in to pinpoint which specific pairs of groups exhibit these significant differences, thereby providing a more detailed understanding of the comparative performance landscape.

The formula for the critical difference (CD) in the Nemenyi test is given by:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{4}$$

The value for q-alpha was acquired from the following function where we got 3.314 from Charles Zaiontz q-table

$$\alpha 0.05, k = 3, df = \infty > 3.314 \tag{5}$$

$$q_\alpha = \frac{3.314}{\sqrt{2}} = 2.343 \tag{6}$$

A. Statistical Analysis

I conducted the Friedman test followed by the Nemenyi post-hoc test to statistically compare the performance of the selected machine learning algorithms in terms of accuracy, F1 score, and training time on the Spambase dataset.

We use the critical value of 6.2 gathered from [Tables for the Friedman Rank Test] to compare to our Friedman statistic value.

The python calculation implementation see Fig-1 for Friedman and Fig-2 for Nemenyi code. If we where to use the manual function provided by P. Flach we would use the following function like this.

$$\bar{R} = \frac{1}{nk}\sum_{ij} R_{ij} = \frac{k+1}{2} = 2 \tag{7}$$

$$n\sum_{ij}(R_j - \bar{R})^2 = 20 \tag{8}$$

$$\frac{1}{n(k-1)}\sum_{ij}(R_{ij} - \bar{R})^2 = 1 \tag{9}$$

$$CD = 2.343\sqrt{\frac{3(3+1)}{6*10}} \approx 1.0478 \tag{10}$$

TABLE I
ACCURACY

Fold	R-Forest	KNN	SVM
1	0.952 (1)	0.803 (2)	0.716 (3)
2	0.965 (1)	0.804 (2)	0.720 (3)
3	0.961 (1)	0.822 (2)	0.735 (3)
4	0.959 (1)	0.824 (2)	0.711 (3)
5	0.948 (1)	0.828 (2)	0.720 (3)
6	0.963 (1)	0.807 (2)	0.739 (3)
7	0.950 (1)	0.789 (2)	0.693 (3)
8	0.959 (1)	0.830 (2)	0.707 (3)
9	0.948 (1)	0.798 (2)	0.702 (3)
10	0.950 (1)	0.791 (2)	0.707 (3)
Avg-Rank	1	2	3
Avg-Score	0.955445	0.809608	0.714844
Std	0.006576	0.015322	0.014159

1) friedmans test: Accuracy: Friedman stat: 20.0, critical value: 6.2,

For Accuracy, the null hypothesis is rejected. There are significant differences among the algorithms. Due to

$$Critical - value : 6.2 < Friedmanstat : 20$$

2) Nemenyi tests: Accuracy: No significant difference between SVM and KNN due to the following function being false.

$$1.0 > 1.0478$$

The difference between SVM and Random Forest is significant in Accuracy due to

$$2.0 > 1.0478$$

No significant difference between SVM and Random Forest due to the following function being false.

$$1.0 > 1.0478$$

TABLE II
F1-SCORE

Fold	R-Forest	KNN	SVM
1	0.939 (1)	0.749 (2)	0.553 (3)
2	0.955 (1)	0.751 (2)	0.560 (3)
3	0.951 (1)	0.763 (2)	0.561 (3)
4	0.947 (1)	0.776 (2)	0.572 (3)
5	0.934 (1)	0.787 (2)	0.566 (3)
6	0.953 (1)	0.744 (2)	0.565 (3)
7	0.936 (1)	0.720 (2)	0.547 (3)
8	0.946 (1)	0.776 (2)	0.566 (3)
9	0.933 (1)	0.745 (2)	0.539 (3)
10	0.934 (1)	0.738 (2)	0.545 (3)
Avg-Rank	1	2	3
Avg-Score	0.942764	0.754913	0.557372
Std	0.008581	0.020325	0.010917

3) Nemenyi tests: F1-score: No significant difference between KNN and Random forest due to the following function being false.

$$1.0 > 1.0478$$

The difference between SVM and Random Forest is significant in Accuracy due to

$$2.0 > 1.0478$$

No significant difference between SVM and KNN due to the following function being false.

$$1.0 > 1.0478$$

TABLE III
TRAINING TIME

Fold	R-Forest	KNN	SVM
1	447.379 (3)	0.822 (1)	331.950 (2)
2	452.919 (3)	0.674 (1)	353.773 (2)
3	448.581 (3)	0.707 (1)	331.661 (2)
4	453.132 (3)	0.760 (1)	316.375 (2)
5	448.271 (3)	0.625 (1)	330.237 (2)
6	443.043 (3)	0.595 (1)	354.743 (2)
7	491.209 (3)	0.659 (1)	333.438 (2)
8	501.934 (3)	0.703 (1)	338.550 (2)
9	493.416 (3)	0.629 (1)	357.687 (2)
10	473.582(3)	0.624 (1)	320.812 (2)
Avg-Rank	3	1	2
Avg-Score	465.346599	0.679708	336.922407
Std	22.508646	0.070067	14.245110

4) Nemenyi tests: Traning time: No significant difference between SVM and Random forest due to the following function being false.

$$1.0 > 1.0478$$

The difference between KNN and Random Forest is significant in Accuracy due to

$$2.0 > 1.0478$$

No significant difference between SVM and KNN due to the following function being false.

$$1.0 > 1.0478$$

IV. RESULTS AND DISCUSSION

The code for Friedman's test will be used to produce the results after each test. But due to the performance being so similar and no difference in rank was seen. The Friedman test results will only be conducted on 1 table. Because the equation is the same with same result on each table.

From these test we could conclude that there is an significant difference in both Accuracy and F1-Score.

The difference between SVM and Random Forest is significant in Accuracy and f1-score (both are the same in terms of rank) due to

$$2.0 > 1.0478$$

But if we compare these test in training time we can see a significant difference between KNN and Random forest. Where KNN is the fastest algoritm and Random forest being the slowest algorithm.

The difference between KNN and Random Forest is significant in Traning time due to

$$2.0 > 1.0478$$

REFERENCES

[1] P. Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data," Cambridge University Press, pp. 106–110, 2012.

[2] MonkeyLearn, "Classification Algorithms," [Online]. Available: https://monkeylearn.com/blog/classification-algorithms/

[3] Mintalb support page "Methods and formulas for Friedman Test" [Online]. Available: https://support.minitab.com/en-us/minitab/20/help-and-how-to/statistics/nonparametrics/how-to/friedman-test/methods-and-formulas/methods-and-formulas/

[4] Charles Zaiontz, "Studentized Range q Table" [Online]. Available: https://real-statistics.com/statistics-tables/studentized-range-q-table/

[5] Louise Martin, Raymond Leblanc, Nguyen Ky Toan, "Tables for the Friedman rank test". Universite du Quebec a Trois-Rivieres [Online]. Available: https://www.jstor.org/stable/3315656metadatainfotabcontents

```
    calculate friedman test
    n = 10  # Number of samples
    k = 3 # Number of algorithms
    sum_squared = sum(ranked_df['RF rank'])**2 + sum(ranked_df['knn_rank'])**2 + sum(ranked_df['svm_rank'])**2
    freidman_stat = (12 / (n * k * (k + 1)) * sum_squared - 3 * n * (k + 1))
    df = n - 1
    critical_value = 6.2
    print("----------------------------------------")
    print(f"friedmans: {freidman_stat}")
    print("----------------------------------------")
    # calcultate Nemenyi test
    alpha = 0.05  # Significance level
    df_inf = 3.314 # for alpha = 0.05 table q range table - https://real-statistics.com/statistics-tables/studentized-range-q-table/
    q_alpha = df_inf/np.sqrt(2)
    print(f"q-alpha {q_alpha}")
    CD = q_alpha * np.sqrt((k * (k + 1)) / (6 * n))
    print(f"Critical Difference (CD) for {test}: {CD}")
    print("----------------------------------------")

    # Hypothesis Testing
    alpha = 0.05  # Significance level
    if critical_value < freidman_stat:
        print(f"For {test}, the null hypothesis is rejected. There are significant differences among the algorithms.")
    else:
        print(f"For {test}, the null hypothesis is not rejected. There are no significant differences among the algorithms.")

    print(f"Test: {test}, freidman_stat: {freidman_stat}, critical_value {critical_value}, Critical diffrence: {CD}")
```

Fig. 1. Friedman test code.

```
# compare the algorithms with critical difference
    svm_rank = rankings['SVM']
    knn_rank = rankings['KNN']
    rf_rank = rankings['Random Forest']
    algorithms = ['SVM', 'KNN', 'Random Forest']
    ranks = [svm_rank, knn_rank, rf_rank]

    for i in range(len(algorithms)):
        for j in range(i+1, len(algorithms)):
            diff = abs(ranks[i] - ranks[j])
            if diff > CD:
                print(f"The difference between {algorithms[i]} and {algorithms[j]} is significant in {test} due to {diff} > {CD}.")
            else:
                print(f"No significant difference between {algorithms[i]} and {algorithms[j]} due to {diff} > {CD} being false.")
```

Fig. 2. Friedman test code.