ARTIFICIAL INTELLIGENCE

(ENCS3340)

Report for Project # 2

Machine Learning Project

Prepared by:

Pierre Backleh #1201296

Mohammad Salem #1200651

Instructor's Name: Yazan Abu Farha

Section: 2

Date: July 13, 2023

Introduction

In this report, we analyze the achieved results and experimental findings for two classifiers: k-Nearest Neighbors (k-NN) and Multi-Layer Perceptron (MLP). The analysis's objective is to decide how well these classifiers perform on a particular dataset taken from the spambase.csv file. The explanation of evaluation measures, confusion matrices for both classifiers, and suggestions for improving the models are all included in the report.

Dataset and Experimental Setup

A dataset collected from the csv file called spambase. The file consists a group of 4601 entries. Each sample is displayed as a single row with 58 column in it. Each row's final number indicates whether the email was considered to be spam (1) or not (0). The features are the first 57 numbers.

Using 30% testing data and 70% training data, the experimental setup involved dividing the dataset into training and testing sets. The k-NN algorithm's "k" parameter specifies how many nearest neighbors will be taken. We implemented the K-NN classifier with k=3 and Euclidean distance as the distance metric.

- In figures 1 below shows the part where we specified the k parameter in the prediction function:

```
91      # Train the k-NN model and make predictions for it
92      model_nn = NN(X_train, y_train)
93      predictions = model_nn.predict(X_test, K)
94      accuracy1, precision1, recall1, f1_1, cmatrix1 = evaluate(y_test, predictions)
95      print("Confusion Matrix for k-NN:")
96      print(cmatrix1)
```

Figure 1 - k parameter

- The percentage of the dataset that will be used to evaluate the performance of the model is represented by the “test_size” parameter in the “train_test_split()” function.

```
83  def main():
84      print("Done By: Pierre Backleh 1201296 & Mohammad Salem 1200651")
85      # Load data from spreadsheet and split into train and test sets
86      features, labels = load_data('spambase')
87      features = preprocess(features)
88      X_train, X_test, y_train, y_test = train_test_split(
89          features, labels, test_size=TEST_SIZE)
```

Figure 2 – “ test_size” parameter

The MLP classifier was set up with two hidden layers: the first layer with 10 neurons and the second layer with 5 neurons. The logistic function (sigmoid) was chosen as the activation function for the Multi-Layer Perceptron classifier. The logistic function converts any real number to a value between 0 and 1.

```
58  def train_mlp_model(features, results):
59      """
60      Given a list of features lists and a list of labels, return a
61      fitted MLP model trained on the data using sklearn implementation.
62      """
63      model = MLPClassifier(hidden_layer_sizes=(10, 5), activation='logistic')
64      model.fit(features, results)
65      return model
```

Figure 3 - Multi-Layer Perceptron classifier

- The "MLPClassifier" is used with two hidden layers. The first one consists of 10 neurons and the second one has 5 neurons. Additionally, the activation function parameter was set to "logistic" to use the sigmoid function.

Evaluation

We used a number of evaluation criteria, such as accuracy, precision, recall, and F1-score, to figure out how well the classifiers performed.

```
68  def evaluate(labels, predictions):
69      """
70      Given a list of actual labels and a list of predicted labels,
71      return (accuracy, precision, recall, f1).
72
73      Assume each label is either a 1 (positive) or 0 (negative).
74      """
75      accuracy = accuracy_score(labels, predictions)
76      precision = precision_score(labels, predictions)
77      recall = recall_score(labels, predictions)
78      f1 = f1_score(labels, predictions)
79      # to print the confusion matrix
80      cmatrix = confusion_matrix(labels, predictions)
81      return accuracy, precision, recall, f1, cmatrix
```

Figure 4 - evaluate function

- The two lists that are passed to the evaluate() function are labels, which contains the actual labels of the data points, and predictions, which contains the predicted labels.
    1. The "accuracy_score()" method from scikit-learn is used to determine the accuracy score. It measures the correctness of the predictions.
    2. The "precision_score()" method is used to calculate the precision score. It shows the classifier's capacity to avoid false positives.
    3. The "recall_score()" function calculates the recall score, commonly referred to as the sensitivity or true positive rate.
    4. The harmonic mean of recall and precision is known as the F1-score. It offers a balanced measurement of precision and recall. The "f1_score()" function is employed to compute this metric.

Finally, the confusion matrix is calculated using the “confusion_matrix()” function.

Results and Analysis

The outcomes in the Figure 6 show that the k-NN classifier did well with the provided dataset. It successfully identified the majority of spam and non-spam emails, achieving a high accuracy rate. The classifier correctly recognized the majority of the predicted spam emails, according to the precision score, which shows that it had a low false positive rate. The recall score indicates that most of the spam emails were successfully classified by the classifier.

```
Confusion Matrix for k-NN:
[[772  62]
 [ 77 469]]
**** 1-Nearest Neighbor Results ****
Accuracy:  0.8992753623188405
Precision:  0.8832391713747646
Recall:  0.8589743589743589
F1:  0.8709377901578459
```

Figure 6 - outcomes for k-NN classifier

```
Confusion Matrix for MLP:
[[788  46]
 [ 39 507]]
**** 2-MLP Results ****
Accuracy:  0.9384057971014492
Precision:  0.9168173598553345
Recall:  0.9285714285714286
F1:  0.9226569608735213
```

Figure 5 - outcomes for MLP Results classifier

- Furthermore, there were 77 false positives (non-spam emails expected to be spam) and 62 false negatives (spam emails predicted to be non-spam) as shown in the confusion matrix in the figure above. In order to ensure that real spam emails are correctly recognized, reducing false negatives is essential.

The outcomes in figure 5 show that the MLP classifier worked exceptionally well on the provided dataset. It classified the majority of spam and non-spam emails with a high degree of accuracy. The classifier had a low false positive rate, according to the precision score, correctly detecting the majority of expected spam emails. It can still be improved to increase accuracy and reduce misclassifications.

Improvements

1. Determine the classifiers' parameters' ideal values, conduct a methodical hyperparameter search. Investigating various k values in k-NN and fine-tuning parameters like learning rate, regularization strength, and the quantity of hidden layers and neurons in the MLP are all part of this. The hyperparameter space can be efficiently searched using methods like random search.

2. Experimenting with alternative classification such as Naïve Bayes and decision tree which can provide valuable insights into the performance and suitability of different models for the given dataset.

Conclusion

In this project, we looked into how the k-NN and MLP classifiers classified spam and non-spam emails. We evaluated how well these models performed using a specific dataset and looked into ways to make them more accurate and efficient.