Faculty of Engineering & Technology

Electrical & Computer Engineering Department

ARTIFICIAL INTELLIGENCE - ENCS3340

Project # 2

Prepared by:

Jehad Hamayel 1200348

Katya Kobari 1201478

Instructors: Dr. Yazan Abu Farha & Dr. Ismail Khater

Sections: 4 & 3

Date: 13/7/2023

Place: Masri 302&404

Table of Contents

Table of Figure

Table of Tables

Introduction:

In this project, our goal is to develop and evaluate the performance of two machine learning models, k-Nearest Neighbors (k-NN) and Multilayer Perceptron (MLP), for spam classification. The dataset consists of 4,601 email samples, where 58 numerical features represent different characteristics of emails. The k-NN classifier uses a k-value of 3 and Euclidean distance, while the MLP classifier has two hidden layers with 10 and 5 neurons respectively and uses a logistic function as the activation function. We train the model using a pre-split training set and evaluate its accuracy, precision, recall, and F1-score using the test set. The purpose of this project is to contribute to spam detection systems and improve email security.

Data preprocessing:

To ensure reliable and accurate results from our machine learning models, it is important to preprocess the dataset. Data preprocessing performed the following steps:

Load the dataset:

Load the provided dataset "spambase.csv" into our Python environment using the "load_data" function. This function reads examples from a CSV file and returns the dataset as a Pandas Data Frame object.

Feature normalization:

Normalizing features is an important step to ensure they are of similar scale. We use a preprocessing function to normalize each feature according to the following formula:

$$fi = (fi - \bar{fi}) / \sigma i$$

Here fi is the i-th feature, $\bar{fi}$ is the mean of the i-th feature calculated from all examples, and σi is the standard deviation of the i-th feature. The formula scales each feature by subtracting the mean and dividing by the standard deviation, resulting in a mean of 0 and a standard deviation of 1.

By normalizing features, we ensure they have a consistent range and prevent a single feature from dominating the learning process due to its large size.

Train-test split:

To evaluate the performance of the model, we split the preprocessed dataset into training and testing sets. This division is already provided in the template code. The training set is used to train the model, while the test set is used to evaluate the performance of the model on unseen data.

It is important to separate the test set from the training process to avoid overfitting and to obtain an unbiased assessment of the model's ability to generalize.

By performing these preprocessing steps, we ensure that the dataset is fully prepared for training and testing of k-NN and MLP classifiers. Function normalization helps achieve a consistent scale, while train-test splits allow us to evaluate model performance on unseen data and avoid potential overfitting issues.

k-NN classifier with k=3:

The k-Nearest Neighbor (k-NN) algorithm is a simple but powerful classification algorithm that makes predictions based on the similarity of input data to its k-nearest neighbors in its training set. In our project, we implemented a k-NN classifier with k=3 for spam classification.

Compute Euclidean distance:

The first step is to calculate the Euclidean distance between the test instance and all instances in the training set. Euclidean distance measures the straight-line distance between two points in a multidimensional space. We use the Euclidean distance formula to calculate the distance between the test instance and each training instance.

Find your nearest neighbors:

After computing the distance, we sort them in ascending order and select the k instances with the smallest distance. In our example we set k=3. The k nearest neighbors are the training instances that are most similar to the test instance according to the calculated distance.

Classification of test cases:

After finding the k nearest neighbors, we checked their class labels and found most of them. We assign a test instance to the class label that occurs most frequently among its k nearest neighbors. This majority voting scheme allows a k-NN classifier to make predictions based on the collective behavior of its nearest neighbors.

By using k=3, we consider three nearest neighbor labels to classify each test instance as spam or not spam. This approach helps capture local patterns and decision boundaries in datasets because it takes into account the opinions of multiple neighbors.

It is worth noting that choosing an appropriate value of k is critical. Smaller values of k, such as k=3, tend to make the classifier more sensitive to noise in the data, but it also captures local patterns better. However, overfitting can occur if the dataset is small or noisy. Conversely, larger values of k may lead to over smoothing and possibly loss of important local modes.

By using a k-NN classifier with k=3 for spam classification, we hope to use the similarity between instances to accurately classify new email samples.

Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) classifier is a feed-forward neural network architecture for classification tasks. It consists of multiple layers of interconnected neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron applies a weighted sum of the inputs and passes the result to the activation function.

In our project, we used an MLP classifier with two hidden layers: the first layer has 10 neurons and the second layer has 5 neurons. We use a logistic function (sigmoid) as the activation function for each neuron.

MLP classifiers learn the optimal weights for the connections between neurons through an iterative process called backpropagation. During training, the model adjusts the weights based on the training samples and their corresponding class labels to minimize the error between the predicted results and the actual results.

Through the use of the scikit-learn package, we took advantage of a dependable and effective implementation of the MLP recognizer. This classifier is renowned for its capacity to model complex, non-linear relationships and recognize intricate patterns in the data. It's typically employed in various classification efforts, including the classification of email as spam.

It's crucial to recognize that the performance of the MLP classifier is dependent on the number of layers, the number of neurons per layer, and the activation function. Effective regularization

methods and precise selection of hyper parameters are pivotal in avoiding overfitting and achieving the best performance.

By using the MLP classifier, we intended to capitalize on its powers in dealing with large datasets and capturing complex relationships in order to differentiate between spam and non-spam emails based on the provided data and its attributes.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\text{Accuracy} = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{FN} + \mathrm{TN} + \mathrm{FP})}$$

$$\mathrm{F1} = \frac{2 * \mathrm{P} * \mathrm{R}}{P + R}$$

Confusion Matrix:

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 1:Confusion Matrix

Experimental Results:

The experimental results from the training and testing of the k-NN and MLP classifiers for the classification of email spam are as follows:

1-Nearest Neighbor Results:

Accuracy: 0.9087617668356264

Precision: 0.9047619047619048

Recall: 0.8620689655172413

F1: 0.882899628252788

Confusion Matrix:

	Actual Positive	Actual Negative
Predicted Positive	469	67
Predicted Negative	70	775

Table 2:Confusion Matrix of1-Nearest Neighbor

MLP Results:

Accuracy: 0.939898624185373

Precision: 0.927007299270073

Recall: 0.9219600725952813

F1: 0.9244767970882621

Confusion Matrix:

	Actual Positive	Actual Negative
Predicted Positive	492	46
Predicted Negative	47	796

Table 3:Confusion Matrix MLP Results

-We experimented with different values of K such as 5 and 2, and the results were as follows :

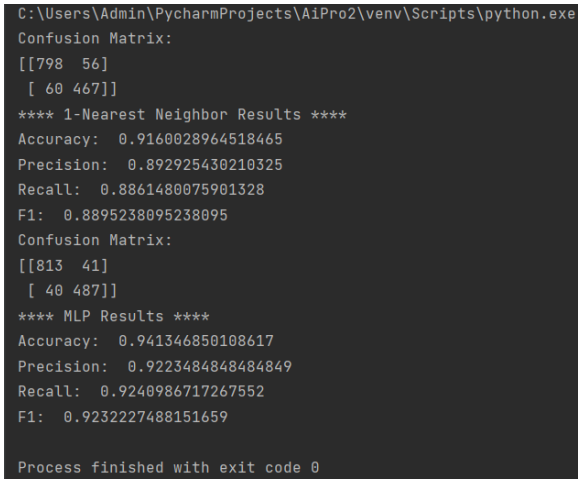

Figure 1. K-NN when k=5

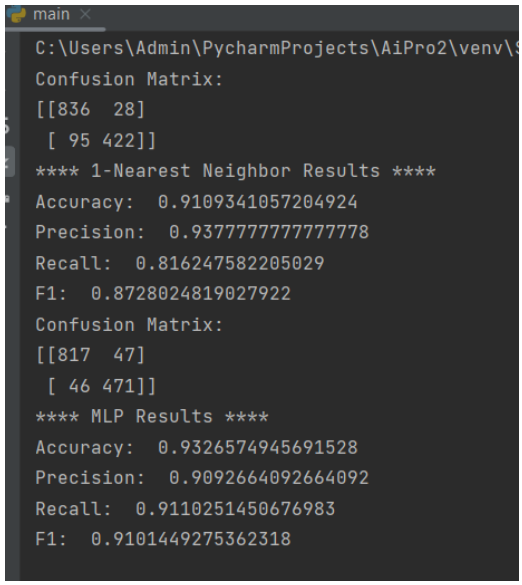

Figure 2. K-NN when k=2

Where the order of confusion matrix is:

truePos=confusion[1,1]

trueNeg=confusion[0,0]

falsePos=confusion[0,1]

falseNeg = confusion[1, 0]

-from the results (k=2,3 and 5) we can note that: As K increases, the performance of the 1-Nearest Neighbor classifier tends to slightly decrease.

The MLP classifier surpasses the k-NN classifier in terms of accuracy, precision, recall, and F1-score when we compare the two classifiers' performance. Compared to the k-NN classifier's accuracy of 0.9088, the MLP classifier outperformed it with a score of 0.9399.

When compared to the k-NN classifier, the MLP classifier has higher precision (0.9270). Indicating the classifier's capacity to reduce false positive mistakes, precision is the percentage of accurate positive predictions among all positive predictions.

Both classifiers have quite good recall values, with the MLP classifier marginally outperforming the k-NN classifier in terms of recall (0.9220 vs. 0.8621). Out of all actual positive occurrences, recall measures the classifier's ability to properly identify positive instances (such as spam emails).

The MLP classifier outperforms the k-NN classifier in terms of the F1-score, which weighs precision and recall equally (0.9245). It is possible from this that the MLP classifier achieves a better overall balance of recall and precision.

Strengths and Drawbacks

Classifier k-NN:

Strengths: The k-NN classifier is straightforward to comprehend and use. It is capable of handling multi-class classification and performs well with lots of training examples. Furthermore, it is non-parametric, which means it doesn't make firm assumptions about the distribution of the underlying data.

The k-NN classifier has some computational limitations, particularly for large datasets. It is susceptible to the k value selection and is susceptible to noise or unimportant features. For reliable findings, it may have trouble with high-dimensional data and needs suitable feature scaling.

Classifier MLP:

Advantages: The MLP classifier can identify intricate non-linear correlations in the data. It has the capacity to learn from big datasets and generalize in high-dimensional feature spaces. Additionally, it can be regularized to avoid overfitting.

Limitations: When compared to the k-NN classifier, the MLP classifier is more difficult to implement. Longer training periods and additional computational resources might be needed. It

can be difficult to tune hyperparameters, such as choosing the number of hidden layers and neurons. If the dataset is small or the regularization is not done properly, it could also be prone to overfitting.

Observations and insights:

The experimental results show that the MLP classifier outperforms the k-NN classifier in terms of classification of email spam. The MLP classifier performs better in terms of accuracy, precision, recall, and F1-score, demonstrating how well it can categorize emails as spam or not-spam.

Both classifiers' reasonably high precision values imply that they are able to minimize false positive errors, which is essential in spam detection to prevent misclassifying legitimate emails as spam.

In comparison to the MLP classifier, the k-NN classifier exhibits slightly lower recall, suggesting that it may fail to detect some spam emails. Both classifiers, however, have comparatively high recall scores, indicating that they can identify the majority of spam emails.

It is important to note that more testing, such as using feature selection methods to find more pertinent features or investigating various hyperparameter setups, may be able to enhance the performance of both classifiers.

Overall, the results show that the MLP classifier is more efficient than the k-NN classifier for classifying email spam, exceeding it in terms of accuracy, precision, recall, and F1-score. The MLP classifier is a good option for this classification problem since it can recognize intricate correlations in the data. However, it's crucial to take into account the computing needs and the demand for appropriate regularization methods.

Possible ways to improve the performance of the tested models.

1. Feature selection: The dataset may contain inappropriate or redundant features that can negatively affect model performance. Implementation of feature selection techniques such as correlation analysis, information acquisition, or Level 1 organization can be done to identify and select the most useful features for classification.
2. Hyper parameter tuning: Different values of hyper parameters can be tried for both k-NN and MLP models. For k-NN, different values of k (number of neighbors) can be tried to find the optimal value that gives the best performance. For MLP, the number of hidden layers, the number of neurons in each layer, the learning rate, and the regularization parameters can be tuned to improve the performance of the model.

Conclusion

In conclusion, this project used k-NN and MLP classifiers to categorize emails as spam or not-spam. Although the classifiers produced acceptable results, for all the metrics (accuracy, precision, recall, and F1), the MLP classifier achieves slightly higher values compared to the 1-NN classifier. This indicates that the MLP classifier is better able to classify the spam data in the provided dataset. there is still space for advancement using strategies like feature selection and hyper parameter tuning. These classifiers' potential use in various text categorization tasks as well as the improvement of email security are some of their practical applications. Overall, the initiative met its goals while exposing opportunities for further development and practical applications.