

**Practical Machine Learning**

**ECEN 478/878**

**Assignment 1**

**Fall 2024**

**Data Exploration and A Study of the k-Nearest Neighbors Model**

ECEN 478: 100 points

ECEN 878: 120 points

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Last Name: Swanson

First Name: Zachary

NUID: 88795124

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Obtained Score:**

**Introduction**

This experiment explores the use of the k-nearest neighbor (kNN) model for binary and multi-class classification problems using the UCI Adult Income and the CIFAR-10 datasets, respectively. The kNN model is an example of an analogy-based learning algorithm. Analogy-based implies that the model makes predictions based on similarities (i.e. analogies) to other instances that the model has seen before.

The kNN model may be considered a simple model as it does not require any explicit training process or parameter estimation. Instead, it makes predictions by directly comparing new instances to the stored training data. When a new data point is introduced, the kNN algorithm calculates the distance between this point and all the other data points in the training set, typically using a metric like Euclidean distance. Based on the closest neighbors, it assigns a class label (for classification tasks) or predicts a value (for regression tasks). This "lazy" approach to learning means that all computation occurs at prediction time, making it straightforward but computationally expensive with large datasets. Additionally, the model's performance is highly dependent on the choice of the distance metric and the number of neighbors (k).

This experiment also explores general topics related to machine learning. These topics include hyperparameter tuning, feature selection, classification performance metrics, and the curse of dimensionality.

**Methodology**

The experiment was conducted in two parts, binary and multiclass classification, each with additional sub-experiments. To perform binary classification on the UCI Adult Income dataset, the data was first preprocessed by loading it into a Pandas DataFrame and exploring its structure. Categorical features were one-hot encoded, and missing values were handled by replacing them with the median value for each feature. After preparing the data, it was split into training and test sets and converted into NumPy arrays for further analysis.

Following preprocessing, several k-NN classification experiments were conducted. In **Experiment 1**, a k-NN model was trained without feature standardization, and key performance metrics such as accuracy, precision, recall, F1 score, and the confusion matrix were recorded. **Experiment 2** applied standardization to the features before training, with the same metrics reported. **Experiment 3** involved generating ROC and precision-recall (PR) curves, identifying the optimal threshold from the PR curve, and evaluating performance based on this threshold.

In **Experiment 4**, Sequential Feature Selection was used to identify the 10 most significant features, which were then used to train a k-NN model. The model's performance was evaluated using the same metrics as in previous experiments. For **Experiment 5** (graduate-level), four different feature subsets were selected based on correlation with the target, and k-NN models were trained on each subset. The dataset was standardized, and hyperparameters were tuned to optimize performance. Metrics such as accuracy, precision, recall, F1 score, and the confusion matrix were reported for each experiment, with hyperparameters (n\_neighbors, p, and weights) adjusted to improve the model's results.

To perform multi-class classification on the CIFAR-10 dataset, the data was first preprocessed. The CIFAR-10 dataset consists of 60,000 32x32 color images across 10 classes, with 50,000 images used for training and 10,000 for testing. The images were flattened from their original dimensions (32x32x3) to a 1D array of 3072 features using the NumPy reshape function. The labels, initially loaded as 1D column vectors, were also converted into 1D arrays using the ravel function. Finally, the feature data was scaled using min-max normalization by dividing each value by 255.0.

In **Experiment 6**, a k-NN classifier was trained on the preprocessed dataset to perform multi-class classification. The model’s training accuracy, test accuracy, and confusion matrix were reported. While hyperparameter tuning using grid search was suggested, due to the high dimensionality of the data and the computational expense, predefined values of n\_neighbors=5 and p=1 were used, with other hyperparameters set to their default values.

**Results**

Use the following two tables in the report. Add more rows, as required.

Table 1 - Classification performance metrics for experiments one through six.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Train Accuracy | Test Accuracy | Test Precision | Test Recall | Test F1 |
| Experiment 1 |  | 1.000 | 0.768 | 0.523 | 0.444 | 0.480 |
| Experiment 2 |  | 1.000 | 0.836 | 0.701 | 0.561 | 0.623 |
| Experiment 3 | Opt thresh = 0.43 | 1.000 | 0.836 | 0.661 | 0.660 | 0.660 |
| Experiment 4 | Non-opt thresh | 0.856 | 0.852 | 0.807 | 0.509 | 0.625 |
| Opt thresh = 0.32 | 0.852 | 0.849 | 0.774 | 0.528 | 0.628 |
| Experiment 5 | Subset 1 features: | 0.855 | 0.844 | 0.714 | 0.593 | 0.648 |
| Subset 2 features: | 0.857 | 0.851 | 0.731 | 0.604 | 0.662 |
| Subset 3 features: | 0.788 | 0.785 | 0.549 | 0.610 | 0.578 |
| Subset 4 features: | 0.854 | 0.841 | 0.712 | 0.575 | 0.636 |
| Experiment 6 |  | 0.535 | 0.38 | 0.46 | 0.38 | 0.37 |

Show the optimal hyperparameters in the following table in the report.

Table 2 - Optimal hyperparameters determined via hyperparameter tuning for experiments one through six.

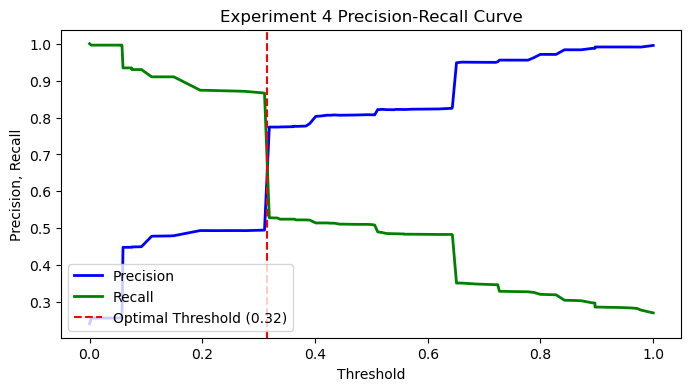
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | n\_neighbors | p | weights |
| Experiment 1 |  | 3 | 1 | distance |
| Experiment 2 |  | 51 | 1 | distance |
| Experiment 3 | Opt thresh = 0.43 | 51 | 1 | distance |
| Experiment 4 | Opt thresh = 0.32 | 87 | 1 | distance |
| Experiment 5 | Subset 1 features: | 17 | 1 | uniform |
| Subset 2 features: | 21 | 2 | uniform |
| Subset 3 features: | 83 | 2 | uniform |
| Subset 4 features: | 17 | 50 | uniform |
| Experiment 6 |  | 5 | 1 | uniform |

**A graph of a curve

Description automatically generated**

**A diagram of a graph

Description automatically generated**

****

**Conclusion**