University of Nebraska Lincoln

Practical Machine Learning
ECEN 478/878

Assignment 1

Fall 2024

Dataset Exploration and A Study of the k-Nearest Neighbors Model

---

Assignment Goals

The goal of this assignment is to explore a dataset and study the k-Nearest Neighbors (k-NN) model for classification.

- **Part A**: Performance optimization via data standardization, feature selection, & varying threshold for a structured dataset
- **Part B**: Understanding the curse of dimensionality & the fundamental limitation of the k-NN model using an unstructured dataset

---

Assignment Instructions

Note: You must use Scikit-Learn to create k-NN models. You are allowed to use Python libraries such as Pandas, NumPy, and Matplotlib.

i. The code should be written in a Jupyter notebook. Use the following naming convention.
   <lastname_1>_<lastname_2>_ assignment1.ipynb
ii. For each experiment, use unique names for your machine learning models.
iii. If you need to partition the data multiple times for training different models, use unique names for each training and testing subset.
iv. The Jupyter notebook should be submitted via Canvas.

The programming code will be graded on **implementation, correctness, and the stly of presentation**.

Score Distribution

ECEN 478: 100 points
ECEN 878: 120 points

Part A: Classification of Structured Data

You will create k-NN classifiers to perform binary classification on the following structured dataset.

Dataset:
The UCI Adult Income Dataset (also known as the "Census Income" dataset) *adult.csv* comprises 14 attributes including categorical and numerical features. The target "income" class is a binary variable (<=50K, >50K). The prediction task is to determine whether a person makes over 50K a year.

Pre-Processing:

- Load the CSV file as a Pandas DataFrame object. Name it "df". Display the first five rows and a summary of dataset information. [**2 pts**]
- Display the number of categories and the list of categories for each categorical variable. For example, for the categorical variable 'sex', display its two categories: 'Male' and 'Female'. [**3 pts**]
- One-hot encode the categorical features and combine the one-hot encoded features with the non-categorical columns. Ensure that only (n-1) columns are added for a categorical variable with n categories. The DataFrame object "df" should include all one-hot encoded and non-categorical features. Finally, display the first five rows of the modified "df". [**5 pts**]
- Display a list of the feature names along with their indices. [**3 pts**]
- Display the first five values of the binary target column (0s and 1s). Create a histogram of the target column. [**4 pts**]
- Identify the features (columns) with missing values in the "df". For each feature with missing values, display the total count of missing values. [**5 pts**]
- Replace the missing values in each feature with the median value. First, compute the median of each feature. Then, replace the missing values with these medians. Display the first five rows of the modified "df". [**6 pts**]
- Compute the Pearson correlation coefficient (also known as the standard correlation coefficient) between the binary target and all features. Display the complete list of correlations. [**2 pts**]
- Create a deep copy of the DataFrame object using the copy() method of DataFrame. Name it as "df_main". You will need it for Experiment 5. [**1 pts**]
- From "df", create separate DataFrame objects for the features and the target. [**2 pts**]
- Convert the feature and target DataFrame objects into NumPy arrays (X for the feature matrix and Y for the target matrix). [**2 pts**]
- Display the shape and data type of the feature and target arrays. [**2 pts**]

- [**required only for graduate students**] Create a bar plot for the new target array to show the distribution of samples in the two classes. You must use NumPy arrays to create the plot and are not allowed to use DataFrame methods. Label the vertical axis as "Number of Instances". Display the class labels ("high" or "low") at the bottom of each bar on the horizontal axis. The title of the figure should be "Distribution of Classes", and the figure dimensions should be (8, 8). **[5 pts]**

- Partition the dataset into training & test subsets: 80% training & 20% test (you may use Scikit-Learn's train_test_split() function) **[1 pts]**

Experiments:
Perform binary classification using k-NN models for the following experiments. Tune hyperparameters to determine the optimal values for the following 3 hyperparameters: *n_neighbors*, *p*, and *weights*.

- Experiment 1) All features & no standardization. Report train accuracy, test accuracy, test precision, test recall, test F1 score, and test confusion matrix. **[5 pts]**
- Experiment 2) All features & apply standardization. Report train accuracy, test precision, test recall, test F1 score, and test confusion matrix. **[6 pts]**

- Experiment 3) Generate the receiver operating characteristic (ROC) curve, area under the curve (AUC) score, and the precision-recall (PR) curve for the model of experiment 2. **[6 pts]**

- Find the optimal threshold from the PR curve. Using the optimal threshold, compute train accuracy, test accuracy, test precision, test recall, test F1 score, and test confusion matrix. **[4 pts]**

- Experiment 4) Use Scikit-Learn's SequentialFeatureSelector class from sklearn.feature_selection to identify the 10 most significant features. Create a new dataset, X_significant, that includes these 10 significant features. Then, build a k-NN model using these features and tune the hyperparameters. *Please note that this process may take a considerable amount of time due to the computational expense of feature selection*. Report the train accuracy, test accuracy, precision, recall, F1 score, and confusion matrix for the k-NN model. You may use/adapt the following starter code. **[6 pts]**

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

# Forward selection: starts with no features and adds one at a time
sfs = SequentialFeatureSelector(knn, n_features_to_select=10, direction='forward')
sfs.fit(X_train, y_train)
```

```
selected_features = sfs.get_support(indices=True)  # Indices of selected features
```

- Experiment 5) [**required only for graduate students**] For this experiment, use the **df_main** DataFrame object that you created in the pre-processing stage. Ensure that its target column is binary-valued and of integer type before using it. You will select at least four subsets (each containing at least ten features) to train k-NN models. You may use Pearson's correlation coefficient to select features. Your goal is to increase the test accuracy. You are free to try any combination of features. **Justify the selection of features for each subset in no more than three lines**.

  Before creating a k-NN model, first, create NumPy feature and target arrays from the DataFrame. Partition the dataset into training and test subsets as before. Standardize the data. Then, perform hyperparameter tuning to find the optimal model.

  This process should be repeated for at least four combinations of features. You may try more combinations. For each combination, show train accuracy, test accuracy, test precision, test recall, test F1 score, and test confusion matrix.

  [15 pts]

Use the following two tables in the report. Add additional rows for Experiment 5 as necessary to include all relevant details.

		Train Accuracy	Test Accuracy	Test Precision	Test Recall	Test F1
Experiment 1						
Experiment 2						
Experiment 3	Optimal threshold =					
Experiment 4						
Experiment 5	Subset 1 features:					
	Subset 2 features:					
	Subset 3 features:					
	Subset 4 features:					

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

		n_neighbors	p	weights
Experiment 1				
Experiment 2				
Experiment 3	Optimal threshold =			
Experiment 4				
Experiment 5	Subset 1 features:			
	Subset 2 features:			
	Subset 3 features:			
	Subset 4 features:			

Answer the following question.

- Q-1) How long did feature selection take in Experiment 4? List the names of the selected features. Did it result in better test performance compared to Experiments 2 and 3? What are your thoughts on the effectiveness of this feature selection process?? **[5 pts]**

Part B: Classification of Unstructured Data

Create a k-NN classifier to perform multi-class classification on the following unstructured dataset.

Dataset:
The CIFAR-10 dataset (Canadian Institute For Advanced Research) contains 60,000 32 x 32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. You may directly load this dataset using the Keras API:
https://keras.io/api/datasets/cifar10/

The train set contains 50,000 images, and the test set contains 10,000 images.

Pre-processing: **[8 pts]**
You will need to perform some pre-processing steps. The first step is to reshape the data. The dimension of the training set is 50000 x 32 x 32 x 3 and test set is 10000 x 32 x 32 x 3.

Before you use this data for the k-NN model, you need to flatten each sample (i.e., 32 x 32 x 3 = 3072) such that the dimension of the training and test set becomes:

- 50000 x 3072
- 10000 x 3072

Use the NumPy "reshape" function for this purpose.
https://numpy.org/doc/stable/reference/generated/numpy.reshape.html

Example: X_train = X_train.reshape((X_train.shape[0], 3072))

The train and test labels will be loaded as a 1D column vector. You need to convert the 1D vector into a 1D array (because the sklearn k-NN model requires the label data to be a 1D array).

You may convert the label data into a 1D array by using the NumPy "ravel" function:
https://numpy.org/doc/stable/reference/generated/numpy.ravel.html

Example: y_train = y_train.ravel()

Finally, scale the data using the min-max scaling technique, i.e., by dividing the training & test data matrix by 255.0.

Example: X_train = X_train/255.0

Experiment:

- Experiment 6) Create a k-NN classifier for multi-class classification. Report the training accuracy, test accuracy, and test confusion matrix.

You are free to perform hyperparameter tuning using grid search. However, note that due to the high dimensionality of the data, training a k-NN model may take **several hours**, and hyperparameter tuning across a range of values could extend **beyond a day**. If you have the time and patience, you may proceed with hyperparameter tuning. Otherwise, use the provided values for the following hyperparameters:

- n_neighbors=5
- p=1

For the remaining hyperparameters use default values.

[14 pts]

Answer the following question.

- Q-2) To answer the questions below, compare the poor performance of your k-NN model on the CIFAR-10 dataset with the performance of a k-NN model on the MNIST handwritten digits dataset. Click on the following link to observe the performance of a k-NN model on the MNIST dataset, which achieved over 97% test accuracy. https://github.com/rhasanbd/k-Nearest-Neighbors-Learning-Without-Learning/blob/master/k-NN-6-Curse%20of%20Dimensionality.ipynb

  However, your k-NN model is expected to perform poorly on the CIFAR-10 dataset.

  a) Explain why your k-NN model struggles to achieve high test accuracy on the CIFAR-10 image classification problem.
  b) Why does a k-NN model perform accurately on the MNIST handwritten digits image classification problem? The following notebooks may be helpful in answering this question: https://github.com/rhasanbd/Study-of-Analogy-based-Learning-Image-Classification
  c) Is it possible to achieve over 90% accuracy on the CIFAR-10 dataset using a k-NN model? Justify your answer.

  [3 + 3 + 2 pts]

Deliverables:

You will submit two artifacts.

- A single Jupyter notebook containing all experiments. For each experiment, display the required performance measures. If necessary, you may split your experiments into two notebooks and submit them separately. Ensure your code is clearly annotated and includes header descriptions for each block. ***Note that if the notebook lacks annotations or has poor annotation, 5 points will be deducted***.
- A **PDF** copy of the written report, including a cover page (available on Canvas). The report must include the following items.

a) Use the two tables given in Part A to report results for Experiments 1 to 5.
b) Experiment 3: display the ROC curve and PR curve.
c) Experiment 6 report: training accuracy, test accuracy, test confusion matrix, values of the following hyperparameters: *n_neighbors*, *p*, and *weights*
d) Answer to Q-1 & Q-2 (3 parts a, b, and c).

The PDF file should be named as follows: <lastname_1>_<lastname_2>_ assignment1.pdf