ECE 443 / ECE 539 (Fall'24) – Programming Exercise #2 (50 points)

Last updated: October 3, 2024

Rationale and learning expectations: In our ongoing studies, the significance of feature engineering and the foundational role of statistical concepts, like the Law of Large Numbers (LLN), in machine learning have emerged as critical themes. This exercise serves as a hands-on exploration, guiding students through the practical application of feature engineering on real-world environmental telemetry data. Furthermore, it emphasizes the nexus between empirical risk minimization (ERM) and LLN in the machine learning landscape. Through this activity, learners are expected to hone their skills in data feature extraction, appreciate the implications of ERM, and grasp the robustness LLN offers to model generalizability when faced with vast datasets.

General Instruction: All parts of this exercise must be done within a Notebook, with text answers (and other discussion) provided in text cells and commented code provided in code cells. Please refer to the solution template for Exercise #2 as a template for your own solution. In particular:

- Replace any text in the text cells enclosed within square brackets (e.g., `[Your answer to 1.1a goes in this cell]`) with your own text using Markdown and/or LaTeX .
- Replace any text in the code cells enclosed within pairs of three hash symbols (e.g., `### Your code for 1.1a goes in this cell ###`) with your own code.
- Unless expressly permitted in the template, **do not** edit parts of the template that are not within square brackets / three hash symbols and **do not** change the cell structure of the template.
- Make sure the submitted notebook is fully executed (i.e., submit the notebook only after a full run of all cells).

Restrictions: You are free to use `numpy`, `pandas`, `scipy.stats`, `matplotlib`, `mpl_toolkits.mplot3d`, `seaborn`, and `IPython.display` packages within your code. Unless explicitly permitted by the instructor, you are not allowed to use any other libraries, packages, or modules within your submission.

Notebook Preamble: I suggest importing the different libraries / packages / modules in the preamble as follows (but you are allowed to use any other names of your liking):

```
import numpy as np
import pandas as pd
from scipy import stats as sps
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from IPython.display import display, Latex
```

1 Feature Engineering for *Environmental Sensor Telemetry Data*

In this part, we will focus on 'feature engineering' (*aka*, hand-crafted features) for the *Environmental Sensor Telemetry Data*. This dataset corresponds to *time-series* data collected using three identical, custom-built, breadboard-based sensor arrays mounted on three Raspberry Pi devices. This dataset was created with the hope that temporal fluctuations in the sensor data of each device might enable machine learning algorithms to determine when a person is near one of the devices. You can read further about this dataset at Kaggle using the link provided below. The dataset has been organized and structured as a `csv` file within Kaggle, which is being provided to you as part of this exercise.

- **Kaggle dataset link:** `https://bit.ly/environmental-sensor-data`
- **Dataset** `csv` **filename:** `iot_telemetry_dataset.csv`

 waheed.bajwa@rutgers.edu

1.1. Load the dataset from the `csv` file and carry out basic exploration of data by addressing the following questions.

(a) (3 points) Based on your reading of the dataset from Kaggle, is this a supervised machine learning task or an unsupervised machine learning task? Justify your answer as much as possible in a text cell.

(b) (1 point) How many data samples are there in the dataset? *Note:* All such questions must be answered programmatically in one or more code cells.

(c) (1 point) How many samples are associated with the device having MAC address `00:0f:00:70:91:0a`?

(d) (1 point) How many samples are associated with the device having MAC address `1c:bf:ce:15:ec:4d`?

(e) (1 point) How many samples are associated with the device having MAC address `b8:27:eb:bf:9d:51`?

1.2. We now turn our attention to preprocessing of the dataset, which includes one-hot encoding of categorical variables and standardization of non-categorical variables that don't represent time. Note that no further preprocessing is needed for this data since the dataset does not have any missing entries.[1]

(a) (3 points) Provide two *Grouped Bar Charts*, with grouping using the three devices, for the means and variances associated with `co`, `humidity`, `lpg`, `smoke`, and `temp` variables. Comment on any observations that you can make from these charts in a text cell.

(b) (6 points) *Standardize* the dataset by making the data associated with `co`, `humidity`, `lpg`, `smoke`, and `temp` variables zero mean and unit variance. Such standardization, however, must be done separately for data associated with each device. This is an important lesson for practical purposes, as data samples associated with different devices cannot be thought of as having the same mean and variance.

(c) (3 points) One-hot encode the categorical variables of `device`, `light`, and `motion`. In this exercise (and subsequent exercises), you are allowed to use `pandas.get_dummies()` method for this purpose.

(d) (1 point) Print the first 20 samples of the preprocessed data (e.g., using the `pandas.DataFrame.head()` method).

(e) (1 point) Why do you think the `ts` variable in the dataset has not been touched during preprocessing? Comment as much as you can in a text cell.

1.3. (5 points) Map the `co`, `humidity`, `lpg`, `smoke`, and `temp` variables for each data sample into the following six independent features:[2]

1. mean of the five independent variables (e.g., use `mean()` function in either `pandas` or `numpy`)
2. geometric mean of the absolute values of the five independent variables. Make sure to take the absolute value of each variable before computing the geometric mean (e.g., use `gmean()` function in `scipy.stats`)
3. harmonic mean of the absolute values of the five independent variables. Ensure you take the absolute value of each variable before computing the harmonic mean (e.g., use `hmean()` function in `scipy.stats`)
4. variance of the five independent variables (e.g., use `var()` function in either `pandas` or `numpy`)
5. kurtosis of the five independent variables (e.g., use `kurtosis()` function in `scipy.stats`)
6. skewness of the five independent variables (e.g., use `skew()` function in `scipy.stats`)

Print the first 40 samples of the transformed dataset (e.g., using the `pandas.DataFrame.head()` method), which has the six features calculated from the five original independent variables.

Remark 1: One of the things you will notice is that there are some terminologies being used in descriptions of the functions in `scipy.stats` that might not be familiar to you. It is my hope that you can try to get a handle on these terminologies by digging into resources such as `Wikipedia`, `Stack Overflow`, `Google`, etc. Helping you become comfortable with the idea of lifelong self-learning is one of the goals of this course.

[1] Be aware that real-world data in most problems is never this nice!

[2] For the geometric and harmonic means, ensure that you're computing them using the absolute values of the variables. This is because these means are only defined for positive numbers.

 waheed.bajwa@rutgers.edu

2 Empirical Risk Minimization and the Law of Large Numbers

Empirical risk minimization (ERM) is a fundamental principle in machine learning that provides the basis for training algorithms using finite samples. At the core of this approach is the assumption that the sample average, obtained from the training data, converges to the expected value (or population mean) as the sample size increases. This is where the Law of Large Numbers (LLN) becomes crucial. The LLN guarantees that the empirical average of a random sample will converge to its expected value as the sample size grows. In the context of machine learning, this implies that by increasing the sample size, our machine learning models' empirical risks (based on training data) become better approximations of their expected risks (over the entire distribution). This is vital for ensuring the generalization of our models to unseen data.

For this exercise, we will delve deeper into understanding this relationship. Through simulations, you will visualize and appreciate how the sample size plays a pivotal role in making the sample average converge to the expected value. This insight provides a foundational understanding of why and how ERM works in the realm of machine learning.

2.1. In this problem, we will be simulating n rolls of a fair six-sided die using the `numpy.random.choice()` function. The primary objective is to compute the empirical average, represented by:

$$M_n = \frac{1}{n}\sum_{i=1}^{n} D_i,$$

where D represents the random variable corresponding to the die value, and D_i is the i-th realization of this random variable.

(a) (3 points) Initiate your sample size at $n = 1$ and extend up to $n = 10,000$. For each sample size in this range, compute the empirical average M_n based on the n samples. Store these averages for the tasks that follow.

(b) (5 points) For each empirical average M_n you computed, determine the absolute discrepancy between this average and the expected outcome for a fair die roll. That is, compute and record $|M_n - \mathbb{E}[D]|$ for every n in the set $\{1, \ldots, 10,000\}$. Remember, you should calculate $\mathbb{E}[D]$ based on the properties of a fair die.

(c) (3 points) Using `matplotlib`, graph the empirical averages against the number of samples n. Ensure the horizontal axis, depicting the sample size (ranging from 1 to 10,000), and the vertical axis, showing the empirical average, are appropriately labeled. Superimpose a horizontal line on the graph at the expected value $\mathbb{E}[D]$ to indicate the expected result of a fair die roll, and ensure you provide a legend. Analyze and discuss the behavior of the empirical average, especially as it approaches the expected value line.

(d) (3 points) In your next step, visualize the absolute discrepancies you calculated in part (b). Once again, the horizontal axis should represent the number of samples n, and the vertical axis should display the absolute discrepancy between the empirical average and the expected outcome $\mathbb{E}[D]$ for each n. Both axes should be suitably labeled. After plotting, discuss your observations within the context of the Law of Large Numbers.

2.2. Building on the concepts from the previous problem, now consider an unfair six-sided die where the probability of rolling a 6 is 0.5, and the probability for each other number (1 through 5) is 0.1. Continue using the `numpy.random.choice()` function for the simulation.

(a) (3 points) As in the previous question, start with a sample size of $n = 1$ and go up to $n = 10,000$. Compute the empirical average M_n for each of these sample sizes using the n rolls of the unfair die. Ensure you retain these averages for subsequent parts.

(b) (5 points) Using the empirical average M_n you obtained, evaluate the absolute discrepancy between this average and the expected outcome for the unfair die roll. Thus, compute $|M_n - \mathbb{E}[D]|$ for every n in the range $\{1, \ldots, 10,000\}$. It's crucial to determine $\mathbb{E}[D]$ considering the biased probabilities provided for this die.

(c) (1 point) Now, using `matplotlib`, plot these empirical averages as a function of the sample size n. The axes should be labeled just as in the previous question. Overlay this plot with a horizontal line at the value of $\mathbb{E}[D]$ for the unfair die. Remember to provide a legend if multiple curves are present. Discuss any trends or behaviors you notice in the empirical average, especially when compared to the expected outcome for the unfair die.

 waheed.bajwa@rutgers.edu

(d) (1 point) As your final task, visualize the absolute discrepancies computed in part (b). The axes should be designed similarly to the previous question. Upon completing this plot, provide insights regarding the results, especially when considering the Law of Large Numbers and the inherent bias in the die.

 waheed.bajwa@rutgers.edu