Statistical Methods in Artificial Intelligence

Assignment 1

Deadline : 25 August 2023 11:55 P.M

Instructor : Prof Ravi Kiran Sarvadevabhatla

August 17, 2023

1 General Instructions

- Your assignment must be implemented in Python.
- Submit your assignment as a folder with a Jupyter Notebook file (.ipynb) and a bash script. You can have other files as you want, but please ensure the notebook is named 1.ipynb and bash script is named eval.sh.
- While you're allowed to use ChatGPT for assistance, you must explicitly declare in comments the prompts you used and indicate which parts of the code were generated with the help of ChatGPT.
- Plagiarism will only be taken into consideration for code that is not generated by ChatGPT. Any code generated with the assistance of ChatGPT should be considered as a resource, similar to using a textbook or online tutorial.
- The difficulty of your viva or assessment will be determined by the percentage of code in your assignment that is not attributed to ChatGPT. If during the viva if you are unable to explain any part of the code, that code will be considered as plagiarized.
- Clearly label and organize your code, including comments that explain the purpose of each section and key steps in your implementation.
- Properly document your code and include explanations for any non-trivial algorithms or techniques you employ.
- Ensure that your Jupyter Notebook is well-structured, with headings, subheadings, and explanations as necessary.
- Your assignment will be evaluated not only based on correctness but also on the quality of code, the clarity of explanations, and the extent to which you've understood and applied the concepts covered in the course.

- Make sure to test your code thoroughly before submission to avoid any runtime errors or unexpected behavior.
- The Deadline will not be extended.
- Moss will be run on all submissions along with checking against online resources.
- We are aware how easy it is to write code now in the presence of ChatGPT and Github Co-Pilot, but we strongly encourage you to write the code yourself.
- We are aware of the possibility of submitting the assignment late in github classrooms using various hacks. Note that we will have measures in place for that and anyone caught attempting to do the same would be give zero in the assignment.
- This is just the first part of the assignment. The second part of the Assignment on **Decision Trees** will be released soon. **Update : Second Part Released**

2 K Nearest Neighbours

The k-Nearest Neighbors (KNN) algorithm is a fundamental and intuitive machine learning technique for classification and regression tasks. Rooted in the concept of similarity, KNN operates by assigning a label or value to an unlabeled data point based on the majority class or the average of its k nearest neighbours in the feature space. In essence, KNN relies on the assumption that similar instances tend to share similar outcomes. This non-parametric algorithm requires no explicit model training. It can adapt to various data distributions, making it particularly useful for simple classification tasks and cases where underlying patterns are not well-defined. However, while KNN offers simplicity and interpretability, it can be sensitive to noise and high-dimensional data, necessitating careful parameter selection and preprocessing.

2.1 Pictionary Dataset

The Dataset to be used can be found here. This is a npy(numpy) file of dataset where each row corresponds to a datapoint and the number of rows is equivalent to the number of examples in the dataset. The first column in each row of the dataset is the game id, the second column is the embeddings generated by ResNet and third column is the embeddings generated by VIT. The fourth column is the label name(ground truth) and the final column is the guess time which you can ignore. You can read more about Resnets and VITs

2.2 Exploratory Data Analysis

Exploratory dataset visualization and analysis form the cornerstone of data exploration in various domains, serving as a critical initial step before diving into more complex data modeling and decision-making processes. This practice involves visually inspecting and comprehending the distribution, relationships, and patterns within a dataset. Its importance lies in its ability to unearth crucial insights, anomalies, and trends that might not be apparent through raw data alone.
Through exploratory visualization, data practitioners can quickly identify outliers, clusters, and potential errors, leading to data cleaning and preprocessing. Furthermore, it aids in feature selection by revealing the impact of different variables on the target outcome. This process allows for the identification of potential correlations and dependencies, which can significantly impact the choice of algorithms and models during subsequent analysis.
In addition, exploratory analysis empowers domain experts to formulate hypotheses and refine research questions, ultimately guiding the direction of the entire analysis. It provides the necessary context for framing questions that would otherwise be overlooked in the absence of visual cues and statistical insights.

2.2.1 Task 1 [10]

Draw a graph that shows the distribution of the various labels across the entire dataset. You are allowed to use standard libraries like Matplotlib.

2.3 KNN Implementation

You must implement the KNN algorithm from scratch using Python classes in this section. This approach encapsulates the KNN model and its methods within a single class, enhancing code readability, reusability, and maintainability. You can easily manage hyperparameters, data preprocessing, and result interpretation by designing the class. Additionally, you can extend the class to incorporate specific functionalities like cross-validation, hyperparameter tuning, or custom distance metrics, aligning the KNN algorithm with your goals.

2.3.1 Task 1 [30]

The KNN algorithm relies on hyperparameters that significantly impact its performance and behavior. One of the crucial hyperparameters is 'k', representing the number of nearest neighbors considered for classification or regression. A smaller 'k' value might lead to a more flexible model, capturing local patterns, while a larger 'k' value could result in a smoother decision boundary, capturing broader trends. Additionally, the choice of distance metric, such as Euclidean, Manhattan, or cosine distance, significantly influences how the algorithm measures similarity between data points. Other hyperparameters may involve weighting schemes, where closer neighbors have higher influence, and

data preprocessing techniques, such as feature scaling, that can affect the algorithm's sensitivity to input attributes. The thoughtful selection of these hyperparameters plays a vital role in achieving optimal KNN model performance for specific tasks.
Create a KNN class where you implement the following: You should not use sklearn for this.

1. Create a class where you can modify and access the encoder type, k, and distance metric (and any required parameter)of the class
2. Return the inference (prediction) when given the above parameters (encoder type, k, and distance metric).
3. Return the validation f-1 score, accuracy, precision, and recall after splitting the provided dataset into train and val subsets. You are allowed to use sklearn metrics for this part.

2.4 Hyperparameter Tuning

2.4.1 Task 2 [30]

Tasks:

1. Find the best (k, encoder, distance metric) triplet that gives the best validation accuracy for a given data split (your choice).
2. Print an Ordered rank list of top 20 such triplets.
3. Plot k vs accuracy given a choice(yours) of any given distance, encoder pair (with a constant data split).

2.5 Testing

In this section, you are supposed to create bash scripts for testing unseen data.

2.5.1 Tasks [10]

Tasks:

1. Create a bash script that when given data from a file in the form of a list of test embeddings and test labels as a npy file exactly in the same format as the dataset file given to you, return the same metrics as discussed in task 2.2.1

Note:

1. The bash script should take the path to the file as an input.
2. It should print out in a table the accuracy, f1-score, recall, and precision.
3. Proper error handling should be present
4. There will be a penalty if this script does not work as expected.

2.6 Optimization

Think about the time complexity during inference for this particular algorithm? What do you think the graph of the time complexity versus number of samples would look like?

2.6.1 Tasks [30]

Tasks:

1. Is it possible to improve the execution time of the program? Hint: Use **Vectorization**.
2. Plot inference time for initial KNN model, best KNN model, most optimized KNN model, and the default sklearn KNN model.
3. plot the inference time vs train dataset size for initial KNN model, best KNN model, most optimized KNN model, and the default sklearn KNN model.

3 Decision Trees

3.1 Data Exploration

The Dataset to be used can be found here. The dataset for this task, is an sythetic dataset containing numerical, categorical and boolean feature variables. It contains 1000 examples and 11 features. We strongly encourage you to visualize and explore the class distribution and other dataset characteristic. The dataset contains personal information of an user along with groundtruth labels which indicate the advertisement related which sector of industry like Electronics, Food, Clothing etc.
The dataset for this task is an **multilabel** dataset, which means there can exist more than 1 label for each example in the dataset

3.2 Decision Tree

A Decision Tree is a versatile and interpretable machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the input space into subsets, guided by a series of binary decisions based on input features. At each node of the tree, the algorithm selects the feature that best separates the data based on certain criteria, such as Gini impurity or information gain, and continues splitting until a stopping condition is met, such as a maximum tree depth or a minimum number of samples in a node. The result is a tree-like structure where each leaf node corresponds to a predicted class label (for classification) or a predicted value (for regression).

3.3 MultiLabel Classification

Multilabel classification is a scenario where an instance can be assigned multiple labels simultaneously. In some cases, a multilabel classification problem might involve predicting multiple labels for each instance.
The **Powerset** formulation is a strategy used in multilabel classification where each unique combination of labels is treated as a separate class. For example, if you have three labels A, B, and C, you would have $2^3 = 8$ possible label combinations: $\{\}, \{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, and \{A, B, C\}$. Each instance is then associated with one of these label combinations.
Let C be the number of classes . The **multioutput** formulation tries to predict a binary vector $b \in \{0, 1\}^C$ for each input where $b_i = 1$ implies the $label_i$ belongs to the groundtruth for the particular input. **Please note that there can be multiple i such that $b_i = 1$**

3.3.1 Tasks[50]

1. Build a **Decision Tree Classifier** Class with the **Powerset** Formulation which can be initialized by the following set of hyperparameters = [”max depth”, ”max features”, ”criterion”]
2. Build a **Decision Tree Classifier** Class with the **MultiOutput** Formulation which can be initialized by the following set of hyperparameters = [”max depth”, ”max features”, ”criterion”]

For this you are allowed to use the inbuilt sklearn decision tree. Note you are expected to follow the standard datascience practices where you sequentially do

1. data visualization and exploration
2. data preprocessing
3. data featurization
4. train val test splitting

3.4 Hyperparamter Tuning

Given the possible set of hyperparameters given by

1. criterion = [gini, entropy]
2. max depth = [3,5,10,20,30]
3. max features = [3,5,7,9,11]

3.4.1 Task[50]

1. Report the Metrics (Accuracy, F1(micro and macro) , Confusion Matrix, Precision , Recall) for all possible triplet of hyperparamters for both **Powerset** and **MultiOutput Setting**.
2. For both of them also rank the top 3 performing set of hyperparamters according to F1 Score.
3. For the best performing model for each approach report the K Fold validation metrics for an appropriate choice of K While you are allowed to use sklearn metrics here , we **strongly** encourage you to write your own code to compute the metrics and if you do use the sklearn metrics, we **strongly encourage you to verify that it is giving the correct outputs**.