Introduction to Machine Learning – Homework Assignment 2

Homework Title: Implementation of a Simple k-Nearest Neighbor (k-NN) Classifier in Python

Due Date: [In Google Classroom]

Objective

The purpose of this assignment is to deepen your understanding of the K-Nearest Neighborhood (k-NN) algorithm by implementing it from scratch using Python. This exercise will help you:

- Grasp the foundational concepts behind instance based classifiers and lazy learning models.
- Develop skills in implementing machine learning algorithms without relying on pre-built libraries.
- Apply your implementation to the "Play Tennis" toy dataset.
- Analyze and interpret the results of your classifier.

Assignment Description

You are required to implement a **K-Nearest Neighborhood (k-NN) classifier** from scratch. You **must not** use any machine learning libraries such as scikit-learn for the K-Nearest Neighborhood (k-NN) implementation. However, you may use libraries for data manipulation and basic operations (e.g., NumPy, pandas).

Your implementation should:

- Handle both **categorical and numerical features** (though the "Play Tennis" dataset primarily contains categorical data).
- Since you will be using distance metrics all your attributes must be numeric. If your dataset includes nominal attributes convert them into numeric using one-hot encoding.
- K-NN is a lazy learner therefore, in the training stage, given the training set in json format, your code will just store the training data in a more appropriate json format for you to read and do distance calculations easier in classification stage and persist in your HDD. Given a new training set, your code must update the model.

- In the classification stage, you will be given a single instance without class label in json format. Your code must read your model (e.g. json files) from hdd, load it to the memory and classify a single instance using your model and output the predicted class label along with all the calculations printed to the screen.
- If in the classification stage a number of instances are given along with their respective class labels (test set), you code must load the model from the HDD to the memory, classify each instance in the test set one by one and compare the predicted class label with the actual class label to calculate accuracy. All the classification operations, predicted class labels vs. actual class labels must be printed to a log file.

Dataset

You will use the **"Play Tennis"** dataset, a commonly used toy dataset in machine learning examples.

Dataset Description:

The dataset contains weather conditions and a target variable indicating whether tennis was played. It includes the following features:

- **Outlook:** Sunny, Overcast, Rain
- **Temperature:** Hot, Mild, Cool
- **Humidity:** High, Normal
- **Wind:** Weak, Strong
- **PlayTennis (Target Variable):** Yes, No

Dataset Instances:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes

```
'Outlook': {'Sunny': 1, 'Overcast': 2, 'Rain': 3},
 'Temperature': {'Hot': 1, 'Mild': 2, 'Cool': 3},
 'Humidity': {'High': 1, 'Normal': 2},
 'Wind': {'Weak': 1, 'Strong': 2}
```

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Assignment Tasks

1. Data Preparation

- **Load the Dataset:**
  - Create the dataset in your program by manually inputting the data or reading from a json file you create.
- **Explore the Data:**
  - Display the dataset in a tabular format.
  - Summarize the dataset (e.g., count of instances per class)

2. Implement the K-Nearest Neighborhood (k-NN) Classifier

  - K is a hyper parameter that you need to get from user.
  - As distance metric you can use Euclidean Distance or Manhattan Distance. Ask user which one to use.

3. Testing the Classifier

- **Test Instances:**
  - Use the same dataset for testing due to its small size.
  - Alternatively, you can perform **Leave-One-Out Cross-Validation**:
    - For each instance in the dataset:
      - Use all other instances for training.
      - Test on the held-out instance.
- **Make Predictions:**
  - Predict the class label for each test instance.
  - Compare the predicted labels with the actual labels.

4. Evaluate the Model

- **Calculate Performance Metrics:**
  - **Accuracy:** The proportion of correctly classified instances.
  - **Confusion Matrix:** Show true positives, false positives, true negatives, and false negatives.
- **Interpret Results:**
  - Discuss the performance of your classifier.
  - Identify any patterns or observations in misclassified instances.

5. Write a Report

Prepare a report that includes:

- **Introduction:**
  - Briefly explain the K-Nearest Neighborhood (k-NN) algorithm.
  - State the objectives of the assignment.
- **Methodology:**
  - Describe how you prepared the data.
  - Explain the implementation details of your K-Nearest Neighborhood (k-NN) classifier.
  - Discuss how you handled any challenges (e.g., zero probabilities).
- **Results:**
  - Present the performance metrics.
  - Include any tables or figures that help illustrate your results.
- **Discussion:**
  - Analyze the results.
  - Suggest possible reasons for any misclassifications.
  - Discuss the limitations of your implementation.
- **Conclusion:**
  - Summarize your findings and what you learned.
- **References:**
  - Cite any resources you used.

Submission Guidelines

What to Submit

1. **Source Code:**
   - All Python source code files.
   - Json files for the datasets and for the model
   - Include a README file with instructions on how to run your code.
   - Ensure your code is well-commented to explain key sections.
2. **Report:**
   - A PDF file named `Surname_Name_K-NN_Report.pdf`.
   - The report should be 3-5 pages, excluding references and appendices.

How to Submit

- **File Naming Convention:**
  - Compress all your files into a single ZIP file named `Surname_Name_K-NN_HW.zip`.
- **Submission Platform:**

- Submit the ZIP file via the google classroom.

Code Requirements

- **Programming Language:**
  - Use Python 3.x for your implementation.
- **Allowed Libraries:**
  - **Permitted:**
    - NumPy
    - pandas
    - matplotlib or seaborn (for plotting, if necessary)
  - **Not Permitted:**
    - scikit-learn (for model implementation)
    - Any other machine learning libraries that provide K-Nearest Neighborhood (k-NN) functionality.

Academic Integrity

- **Original Work:**
  - The code and report must be your own work.
- **Collaboration Policy:**
  - Discussions with classmates are allowed for understanding concepts.
  - Sharing of code or written content is prohibited.
- **Plagiarism:**
  - Any form of plagiarism will result in a zero for the assignment and may lead to further disciplinary action.

Resources

- **Lecture Materials:**
  - Review the slides and notes on K-Nearest Neighborhood (k-NN) classifiers provided in class.
- **Textbooks:**
  - *Introduction to Machine Learning* by Ethem Alpaydin
  - *Pattern Recognition and Machine Learning* by Christopher Bishop
- **Online Tutorials:**
  - Articles and tutorials on K-Nearest Neighborhood (k-NN) algorithms.
- **Documentation:**
  - Python documentation for libraries like NumPy and pandas.

Questions and Clarifications

- Please ask all your questions from the google classroom only. Add a comment to this assignment

---

Good luck with your assignment! Implementing algorithms from scratch is a valuable exercise that enhances your understanding and prepares you for more advanced topics in machine learning.