Homework 3 – Intro to learning

Administrations

General Guidelines:

- Due Date: 27/01/2022. It can be submitted after the due date if a fine is paid, as per the delay procedure specified in the webcourse.

- The exercise solution should only be submitted in pairs, although you can request permission to submit it individually.

- Exercise solutions must be typed, i.e. not had written.

- Questions about the exercise should be sent to Muhammad Dahamshi with the title AI_HW3 (muhammad.dah@campus.technion.ac.il), It is better to use the piazza of the course. Before asking a question, please check if it has been asked in the past.
piazza Link: piazza.com/technion.ac.il/winter2022/236501

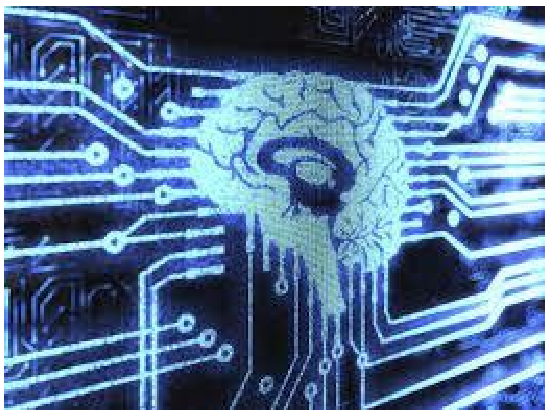
- During the exercise, there may be updates. Updates are required (please check the piazza).

- In Part D you will be asked to invent a feature selection algorithm, as a result, there is a degree of freedom in this part, the score on these parts will be given according to the quality of the solution, both in terms of creativity and in terms of empirical results. The solution must be clearly, formally, and accurately described.

- For the wet part is provided Skelton of the code (that you are required to complete the parts marked TODO)

- You are allowed to use the following python packages: sklearn, pandas ,numpy, random, matplotlib, argparse, abc, typing, all the built-in packages in python, But do not use the learning algorithms, or any other algorithm or data structure that is part of a learning algorithm that you will be asked to implement.

- All changes in the exercise since the first version are marked with an **brown marker**.



Introduction:

Exercise objectives:

- ❖ Understanding the effect of features and examples on the performance of learning algorithms.
- ❖ Understanding the process of parameter tuning and evaluating the performance of learning algorithms.
- ❖ This exercise deals with a binary classification problem. Throughout the exercise, we will experiment with building different types of classifiers (decision tree, KNN) and their extensions. We will deal with the learning of decision trees and feature selection. In this exercise, you will be asked to run a number of experiments and analyze their results. Please perform an in-depth and detailed analysis of the results and attach them to the report.

Before starting, it is highly recommended that you review the relevant lecture and recitation slides.

DATASETS – Overview:

In this exercise we will deal with two $data - sets$, the data is divided into two sets: training set within the $train.csv$ file, and test set within the $test.csv$ file. In general, the training set will be used by us to train the classifiers, and the test set will be used to evaluate their performance.

For your convenience you can use the following auxiliary functions:
load_data_set, create_train_validation_split, create_train_validation_split, get_dataset_split in the utils.py file which load / divide the data into np.array arrays conveniently (please review the documentation of functions).

1. **For the part of ID3:** the data contains metrics collected from photographs designed to distinguish between a benign tumor and a malignant tumor. Each sample contains 30 such metrics and a binary label that determines the type of tumor (0 = benign, 1 = malignant). The first column indicates whether the person is sick (‘M’) or healthy (‘B’).

2. **For the KNN part and feature selection:** The purpose of this data-set is to diagnose whether the patient has diabetes or not, based on certain diagnostic measurements included in the database. The following features have been provided to help us determine if a person has diabetes or not:
    - **Pregnancies:** Number of times pregnant.
    - **Glucose:** Plasma glucose concentration over 2 hours in an oral glucose tolerance test.
    - **BloodPressure:** Diastolic blood pressure (mm Hg).
    - **SkinThickness:** Triceps skin fold thickness (mm).
    - **Insulin:** 2-Hour serum insulin (mu U/ml).
    - **BMI:** Body mass index (weight in kg/(height in m)2).
    - **DiabetesPedigreeFunction:** Diabetes pedigree function (a function which scores likelihood of diabetes based on family history).
    - **Age:** Age (years).

    Outcome: Class variable (0 if non-diabetic, 1 if diabetic)

Part A - Getting to Know The Code

Understanding this part well is important for the rest of the exercise.

KNN-dataset / ID3-dataset Folders:

These folders contain the datasets files for ID3 and KNN, respectively. The data is divided for you into two files: train.csv for the training set and test.csv for the test set.

Utils.py file:

This file contains useful auxiliary functions throughout the exercise, such as loading a dataset and calculating accuracy.

Complete the implementation of the functions $accuracy, l2\_dist$. Read the documentation of the functions and the comments under the TODO comment.

You are provided with unit_test.py as a basic test file that will help you test your implementation.

DecisionTree.py file:

This file contains 3 useful classes for building our ID3 tree.

Class Question: This class implements the branching of a node in a tree. When it retains the feature and value by which we split our data.

The DecisionNode class: This class implements a node in the decision tree. The node contains a Question instance and the two childs true_branch, false_branch where true_branch are the part of the data that answers True to the node question (the match method of the Question class returns True) and false_branch in a similar way.

Class Leaf: This class implements a leaf node in the decision tree. The leaf contains for each of the classes in the data the number of examples of the leaf for each class (i.e. {'B': 5,' M': 6}).

ID3.py / ID3_experiments.py files:

This file contains the ID3 class, see the comments and documentation of the methods. ID3 experiments file contains the experiments (which we will explain later):

basic_experiment , cross_validation_experiment

KNN.py / KNN_experiments.py / KNN_CV.pyc files:

This file contains the implemented KNN class, see the comments and documentation of the methods. And the KNN experiments file which we will implement in part D (features selection).

KNN_CV.pyc file: runs cross-validation for KNN (detailed explanation in part D)

✍️ Part B - Dry Part (30 pts)

Question 1: (8 points)
True/False: Given any data with continuous attributes and binary labels, divided into a training and test group, activation of the MinMax normalization function learned in recitation on the data does not affect the accuracy of the ID3 classifier which learned on the training set and then tested on the test set.

[Answer length is limited to 20 lines]

Question 2: (16 points)
Given data set $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ where n examples are labeled with binary classification $y_i \in \{0,1\}$. Each example consists two continuous attributes $x_i = (v_{1,i}, v_{2,i})$. Assume that there is a goal classifier $f(x): R^2 \to \{0,1\}$ that we are interested to find (it is not known to us) which that the examples in D are consistent with this goal classifier (i.e. there are no noisy examples in D).

In the following sections, for KNN assume a Euclidean distance as the distance function. Also, assume that if there are points in space such that there are several examples at the same distance, first consider examples with a maximum value of v_1 and in case of equality with a value of v_1, first consider examples with a maximum value of v_2. Assume that there are no completely identical examples (i.e. with both the same v_1 value and the same v_2 value).

For the ID3 classifier, it should fit any conditions we have presented for your implementation.

In each section, present a case that meets the conditions presented in the section, explain in words, and attach a graphic description (drawing) that describes the case (which includes at least a description of the goal classifier and the examples you have chosen). Mark positive examples with the + sign (plus) and negative examples with the sign - (minus). A trivial purpose classifier must not be presented, that is, all of the examples are classified as positive or all of the examples as negative.

[3 lines per section, no restriction on the graphs]

a. (4 points)
Present a goal classifier $f(x): R^2 \to \{0,1\}$ and training set with a maximum of 10 examples so that ID3 tree learning will yield a classifier that answers correctly for each possible test sample (i.e. the goal classifier will be learned), but KNN learning will yield a classifier for which there is at least one sample on which he will wrongly classify, for each value K chosen.

b. (4 points)
Present a goal classifier $f(x): R^2 \to \{0,1\}$ and training set with a maximum of 10 examples so that KNN learning will yield a classifier which answers correctly for each possible test sample (i.e. the goal classifier will be learned), but ID3 tree learning will yield a classifier for which there is at least one sample on which he will wrongly classify.

c. (4 points)
Present a goal classifier $f(x): R^2 \to \{0,1\}$ and training set with a maximum of 10 examples so that KNN learning, <u>for a specific K value</u>, will yield a classifier for which there is at least one sample on which he will wrongly classify, <u>and ID3 tree</u> learning will yield a classifier for which there is at least one sample on which he will wrongly classify.

d. (4 points)
Present a goal classifier $f(x): R^2 \to \{0,1\}$ and training set with a maximum of 10 examples so that learning KNN, <u>for a specific K value</u>, will yield a classifier which answers correctly for each possible test sample (i.e. the goal classifier will be learned), <u>and also ID3 tree learning</u> will yield a classifier which answers correctly for each possible test sample.

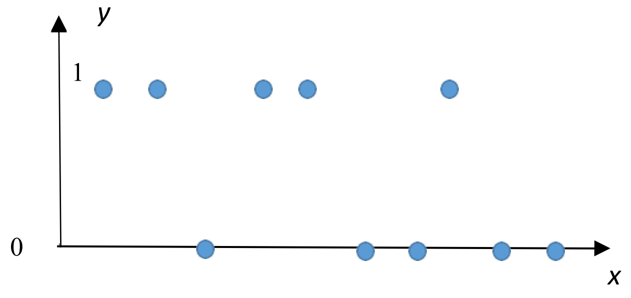
<u>Question 3:</u> (6 points)

Suppose you are using a Majority Classifier on the following training set containing 10 examples where each example has one real-valued feature, x, and a binary class label, y, with value '0' or '1'. Define this Majority Classifier to predict the class label that is in the majority in the training set, regardless of the input value. In case of ties, predict class '1'.

For example: for the training set $\{(1,'1'),(2,'0'),(3,'1')\}$ the Majority Classifier classification is:

$Predict\ (x = 5) = '1'$ because the majority of the labels in the training set is $'1'$.
$Predict\ (x = 2) = '1'$ even though there is sample with 2 and label $'0'$ in the training set.

Answer the question below based in the training set in the following graph:



(a) [3] What is the *training set accuracy*?

(b) [3] What is the *2-fold Cross-Validation accuracy*? Assume the leftmost 5 points (i.e., the 5 points with smallest *x* values) are in one fold and the rightmost 5 points are in the second fold.

Part C – ID3 - Wet Part (60 pts)

Question 4: (5 points)
Complete the implementation of the functions $accuracy, l2\_dist$. Read the documentation of the functions and the comments under the TODO comment.
(Run the appropriate tests in the unit_test.py file to make sure your implementation is correct)

Question 5 - ID3 learning: (30 points)

a. Implement the ID3 algorithm as taught in the lecture.
   Note that all features are continuous. You are asked to use the method of **dynamic auto-discretization** described in the lecture. When considering a threshold value for splitting a continuous attribute, examples with a value equal to the threshold value belong to the group with the values greater than the threshold value. In case there are some optimal attributes at a particular node select the attribute with the maximum index. The implementation should appear in the ID3.py file (complete the missing code after reviewing and internalizing the DecisionTree.py file and the classes it contains).
b. Complete the implementation of the basic_experiment method in the file ID3_experiments.py and run the corresponding part in main function.

✍ specify in the report the accuracy you received.

Question 6 - ID3 Early pruning: (25 points)

Splitting a node takes place as long as it has more examples greater than **M**, that is, in the process of building the tree, "early pruning" is performed, as you learned in the lectures. Note that this means that the trees studied are not necessarily consistent with the examples. Upon completion of the learning (of a single tree), the classification of a new sample using the tree learned is done according to most of the examples of its corresponding leaf.

a. Explain the importance of pruning in general and what it is trying to handle?
   [Answer length is limited to 3 lines]
b. Implement the early pruning as defined in the lecture. The M parameter indicates the minimum number of samples to make a classification. The implementation of the early pruning must also be within the ID3 class located in the ID3.py file)
c. **Bonus** (5 points for the assignment grade) Implementing the hyper-parameter M tuning:
   I. Select at least five different values for parameter M.
   II. For each M value, calculate the accuracy of the algorithm by K-fold cross-validation on the training set only. To divide the training group into K groups use the sklearn.model_selection.KFold function with the parameters n_split = 5, shuffle = True, and random_state equal to your ID number. (Any hyperparameter tuning in the exercise will be done in a similar way).
   ✍ III. Use the results you obtained to create a graph showing the effect of parameter M on accuracy. Attach the graph to the report. (For your use the util_plot_graph function inside the utils.py file).
   ✍ IV. Explain the graph you obtained. For which parameter M did you get the best result and what is this result?

The hyperparameter tuning implementation in the cross_validation_experiment (read the documentation and complete it) in the ID3_ experiments.py file.

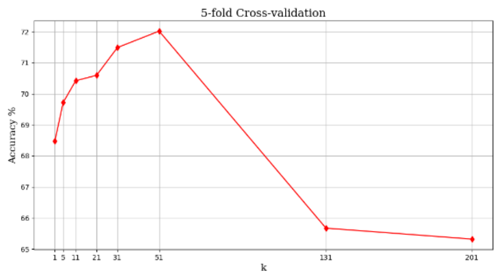
d. Use the ID3 algorithm with the early pruning to learn classified from each training group and make a prediction on the test set. Use the optimal M value you found in section c. (Complete the implementation of the functions best_m_test located in ID3_experiments.py and run the corresponding part in main)
✍ specify in the report the accuracy you received. Did the pruning improve the performance relative to running without pruning in question 5?
Note: if you didn't implement cross-validation in part C, choose M =50.

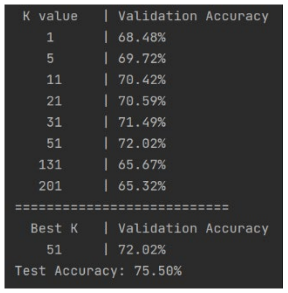
Part D - Wet Part - Features selection (30 pts)

We would like to improve the learning algorithm k-nearest neighbors which implementation is provided to you in the KNN.py file, the KNNClassifier class has two main functions: fit (X, Y) which performs training, and predict(X) which performs the prediction.
In addition, you are provided with a KNN_CV.pyc file that contains bytecode - various instructions from the interpreter.

Explanation: pyc files are created by the Python interpreter when a .py file is imported. They contain the "compiled bytecode" of the imported module/program so that the "translation" from source code to bytecode (which only needs to be done once) can be skipped on subsequent imports if the .pyc is newer than the corresponding .py file, thus speeding startup a little. But it's still interpreted. Once the *.pyc file is generated, there is no need of *.py file, so we hide from you the implementation of the k-cross-validation for the KNN classifier.
When running the KNN_CV.pyc file, the cross-validation which in its run we get the best K hyperparameter.

Question 7: (5 points)
Run the run_cross_validation function located in the KNN_ experiments.py file, the chosen values for the parameter K are [1,5,11,21,31,51,131,201]. Make sure that the result obtained and the accuracy graph of the classifier are:





```
 K value   | Validation Accuracy
    1      | 68.48%
    5      | 69.72%
    11     | 70.42%
    21     | 70.59%
    31     | 71.49%
    51     | 72.02%
   131     | 65.67%
   201     | 65.32%
===========================
  Best K   | Validation Accuracy
    51     | 72.02%
Test Accuracy: 75.50%
```

It can be seen that the value of the best parameter, according to the accuracy metric, with Euclidean distance, Is K = 51, we will use this value throughout the rest of the exercise.
✍ Briefly explain how the KNN algorithm works, mention 2 Cons and 2 Pros of the algorithm.

[Answer length is limited to 5 lines]

Features selection (10 pts + Bonus)

In this section we will focus on the problem of selecting a subset of features, given the full set of properties of size D: $S = \{x_1, \ldots, x_D\}$ we would like to select from these properties only a subset of properties of size b (1≤b≤D) that will be used by us in constructing the classifier.

The reasons why we would like to select a subset of features are varied, here are some of the reasons:

- ❖ Deletion of unnecessary features, i.e. those that are unrelated and do not contribute information for separation between the different classes, or alternatively features that can be expressed by other characteristics in the group. Therefore, they do not provide us with relevant information and can even harm the classified's performance.
- ❖ Providing insights into the classification problem: Selecting the features will give us information about the relevance of the features to the classification process.
- ❖ Reducing computational complexity.

Question 8: (5 points)

✍ We will define the set of all the subsets of S as $P(S) = \{X | X \subseteq S\}$, that is, all possible subsets of size smaller than or equal to D.

a. What is the size of $P(S)$?
b. What is the number of all the subsets of S with fixed size b?

✍ **Question 9:** (**Bonus**: 3 points for the final grade of the course)

In question 8 we saw that the number of possible subsets of features is not polynomial in group size. In general, finding the optimal subset of properties is a difficult problem (NP-hard).

We would like to develop a way (algorithm) to select a subset of good features in a smart way. Which takes the training set x, y and performs preprocessing to select the important features.

Complete the implementation get_top_b_features which takes the parameters x, y, b, k and returns a set of indexes (sorted) of the features we want to select. please review the documentation of the function and complete it.

The KNN classifier will be re-trained on the new training group x', which is filtering group x to the set of properties we got from get_top_b_features.

Note: You are asked to propose an algorithm for feature selection, hence in this section, there is a degree of freedom. The score on this question will be given according to the quality of the solution, both in terms of creativity and in terms of empirical results. The solution must be clearly, formally, and accurately described.

Thought direction: Try to use the tools we have accumulated so far in the course (other solutions will be accepted as well).

[The brute force approach which examines all subsets of size b will not be accepted]

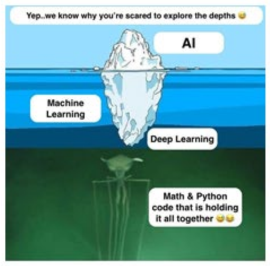
a. On which set (validation/test/training set) should the performance be tested? explain.
b. What is the optimal subset size of you get? did you get an improvement in performance by applying feature selection?
c. Explain in depth the algorithm you performed.

Submission Instructions

• The ID numbers of the two submitters must be written in the report.
• An automated testing system will check your code thoroughly, so please consist with the required format.
• The report should be typed and in PDF format. Reports in word format will be fined.
• The report should be legible and easy to understand.
• Answers in the report should follow the order of the questions.
• You should submit a zip file with the name AI3_<id1>_<id2>.zip (with your ID numbers).
• The zip should contain:
  - The report is in PDF format, named AI_HW3.PDF.
  - All the code files and folders you received:
    - The utils.py file
    - In the ID3 part: ID3.py, ID3_experiments.py
    - In the k-nearest neighbors and feature selection part: KNN_experiments.py
    - Any auxiliary code you implemented in the exercise.

Good Luck! 😊

People with no idea about AI, telling me my AI will destroy the world
Me wondering why my neural network is classifying a cat as a dog..
Dog


Yep..we know why you're scared to explore the depths 😅
AI
Machine Learning
Deep Learning
Math & Python code that is holding it all together 😅😅