COMPUTATIONAL INTELLIGENCE
ASSIGNMENT REPORT

Shivansh Kumar
18100053,CSE
shivansh18100@iiitnr.edu.in
Assignment link: https://github.com/shivansh2310/CI-Assignment

Experiment 1:

Breast cancer cell classification using K-Nearest Neighbor classifier. Use the dataset of file **wisc_bc_data.csv**. Use the following settings to design the classifier:

a. Min-max feature normalization.
b. Out of 569 data samples use the 1 to 469 for creating training dataset. Use rest of the samples to estimate the accuracy of the classifier.

Calculate the accuracies for K = 9, 11, 13, 15, 17 and 19

Experiment Name: Breast cancer cell classification using K-Nearest Neighbor classifier.

Software Used: python 3.8 (libraries used: pandas, numpy, scikitlearn, matplotlib, seaborn)

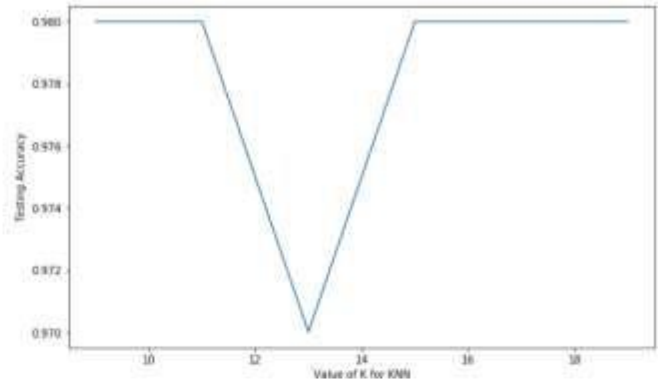
Algorithm of KNN:

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
    i) Calculate the distance between the query example and the current example from the data.
    ii) Add the distance and the index of the example to an ordered collection
    iii) Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
4. Pick the first K entries from the sorted collection
5. Get the labels of the selected K entries
6. If regression, return the mean of the K labels
7. If classification, return the mode of the K labels

Results:

K : Accuracy

```
{9: 0.98, 11: 0.98, 13: 0.97, 15: 0.98, 17: 0.98, 19: 0.98}
```



Best Accuracy obtained during training was for K = 8
(0.99)

Experiment 2:

Breast cancer cell classification using **Weighted K-Nearest Neighbor classifier**. Use the dataset of file **wisc_bc_data.csv** and following settings to design the classifier:

a. Min-max feature normalization.
b. Randomly select 100 healthy and 100 cancerous cell samples to construct the training dataset. Use rest of the samples to estimate the accuracy of the classifier.
c. Calculate the accuracies for K = 9, 11, 13, 15, 17 and 19

Experiment Name: Breast cancer cell classification using **Weighted K-Nearest Neighbor classifier**.

Software Used: python 3.8 (libraries used: pandas, numpy, scikitlearn, matplotlib, seaborn)

Algorithm of Weighted KNN:

The intuition behind weighted kNN, is to give more weight to the points which are nearby and less weight to the points which are farther away. Any function can be used as a kernel function for the weighted knn classifier whose value decreases as the distance increases. The simple function which is used is the inverse distance function.

Algorithm

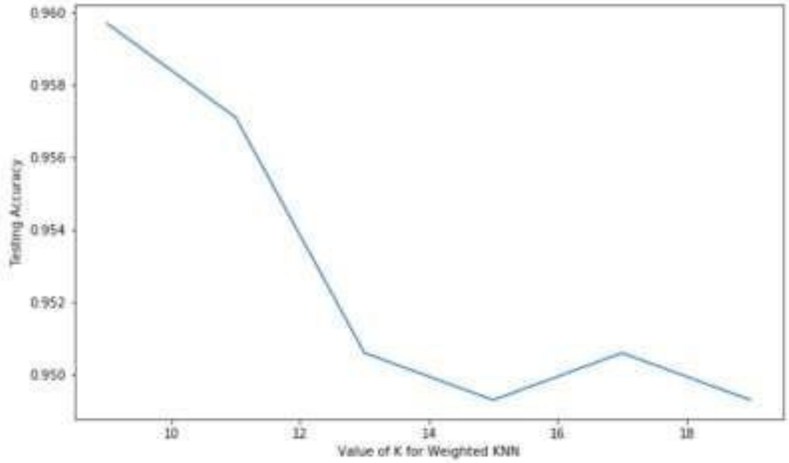
i) Let L = { (xi , yi) , i = 1, . . . ,n } be a training set of observations xi with given class yi and let x be a new observation(query point), whose class label y has to be predicted.
ii) Compute d(xi, x) for i = 1, . . . ,n , the distance between the query point and every other point in the training set.
iii) Select D’ ⊆ D, the set of k nearest training data points to the query points
iv) Predict the class of the query point, using distance-weighted voting. The v represents the class labels. Use the following formula:

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i)$$

Results:

K : Accuracy

{9: 0.9713914174252276, 11: 0.9674902470741222, 13: 0.9622886866059818, 15: 0.9609882964889467, 17: 0.953185955786736, 19: 0.9570871261378413}



Experiment 3:

Write a program to generate 100 sample points which follows the following linear equation: $y = 5x + 3$.
Assume that the x variable take the values in range [0, 10]. Now add the additive white Gaussian noise N(0,1) with zero mean and unity variance to y to obtain $\hat{y} = y + N(0,1)$. Now estimate the values of slope and intercept using the linear regression analysis.

Experiment Name: Estimating values of slope and intercept using Linear Regression

Software Used: python 3.8 (libraries used: pandas, numpy, scikitlearn, matplotlib, seaborn)

Algorithm for Linear Regression:

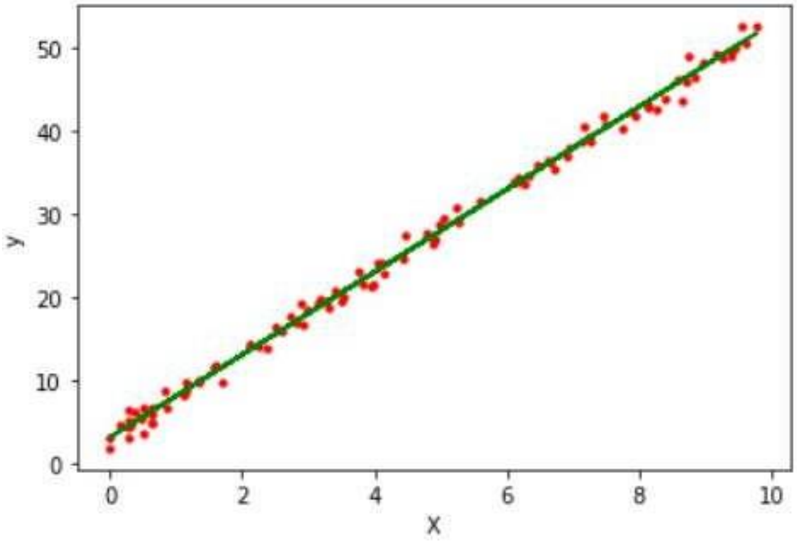
Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It's used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog).

Steps:

a) Find mean of given dependent and independent variables
b) Calculate cross deviation and deviation about independent variable x
    i) SS_xy = sum(y*x) - n*m_y*m_x
    ii) SS_xx = sum(x*x) - n*m_x*m_x
c) Finally calculate regression coefficients estimate
    i) slope = SS_xy / SS_xx
    ii) intercept = m_y - slope*m_x

Results:

The estimated equation found using LR is: Y = 4.94586565548394 *X + 3.356506160339258



Experiment 4:

Classify the mushrooms into poisonous and non-poisonous class using the **Decision Tree classifier**. Use the dataset named **“mushrooms.csv”** and use the following classifier settings to evaluate the performance of classifier:

a. Use 80% of the data sample to train the classifier and rest of them to evaluate the classifier performance.
b. Calculate the accuracy, sensitivity, specificity, true positive rate and false positive rate.

Experiment Name: Classifying the mushrooms into poisonous and non-poisonous class using the **Decision Tree classifier**.

Software Used: python 3.8 (libraries used: pandas, numpy, scikitlearn, matplotlib, seaborn)

Algorithm for Decision Tree Classifier:

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node**.**

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contains possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

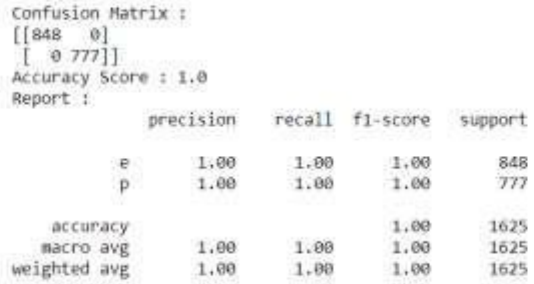
Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

i) Information Gain
Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)
ii) Gini Index
Gini Index= 1- ∑jPj^2

Results:

- Accuracy: 1.0
- Sensitivity: 1.0
- Specificity: 1.0
- True positive rate: 100%
- False positive rate: 100%



```
Confusion Matrix :
[[848   0]
 [  0 777]]
Accuracy Score : 1.0
Report :
              precision    recall  f1-score   support

           e       1.00      1.00      1.00       848
           p       1.00      1.00      1.00       777

    accuracy                           1.00      1625
   macro avg       1.00      1.00      1.00      1625
weighted avg       1.00      1.00      1.00      1625
```