1. Optimization function for SVM().
   1. We have to find the hyperplane that separates the two classes class 1 and class 2 with the maximum margin.
      1. The hyperplane is represented by the equation: w^T x + b = 0, where w is the weight vector and b is the bias term.
      2. We want to maximize the margin subject to the following constraints:
         1. For all x belonging to class 1, w^T x + b >= 1
         2. For all x belonging to Class 2, w^T x + b <= -1
      3. The margin is given by the distance between the hyperplane and the closest point from each class.
      4. Let x1 and x2 be the closest points to the hyperplane from classes class1 and class2, respectively.
      5. The margin is given by the distance between x1 and x2 divided by ||w||.
      6. We want to maximize the margin, which is equivalent to minimizing ||w|| subject to the constraints in step ii.
      7. The optimization is given as:
         1. minimize: ||w||^2
         2. subject to:
            1. For all x belonging to C1, w^T x + b >= 1
            2. For all x belonging to C2, w^T x + b <= -1
      8. The Lagrangian for this problem is given by:
         1. L(w, b, α) = 1/2 ||w||^2 - ∑i=1 to n αi(yi (w^T xi + b) - 1)
         2. where αi is the Lagrange multiplier for the ith constraint.
      9. The dual of the Lagrangian is given by:
         1. maximize: ∑i=1 to n αi - 1/2 ∑i,j αi αj yi yj xi^T xj
         2. subject to: ∑i αi yi = 0 αi >= 0 for all i
      10. Once we have found the optimal values for αi, we can compute the weight vector w as:
          1. w = ∑i=1 to n αi yi xi
      11. The bias term b is computed below:
          1. b = (1/yk) - w^T xk
          2. where xk is any point on the hyperplane and yk is its corresponding class label.
2. SVM solver
   1. Implemented a linear SVM using Scikit-Learns library. Also, all the plots are available in Results folder.
      1. The classification accuracy with various C value are as follows:
         1. C=1
            1. Train\_accuracy=0.69
            2. Test\_accuracy=0.66
            3. Here both train and test accuracy are close enough
            4. Support Vectors:

There are 63 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.14377853 0.11001499 0.75 4.46599757]

* + - 1. C=2
         1. Train\_accuracy=0.69
         2. Test\_accuracy=0.66
         3. Here both train and test accuracy are close enough
         4. Support Vectors:

There are 63 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.14377853 0.11001499 0.75 4.46599757]

* + - 1. C=4
         1. Train\_accuracy=0.69
         2. Test\_accuracy=0.66
         3. Here both train and test accuracy are close enough
         4. Support Vectors:

There are 63 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.14377853 0.11001499 0.75 4.46599757]

* + - 1. C=6
         1. Train\_accuracy=0.74
         2. Test\_accuracy=0.66
         3. Here both train accuracy increased but testing accuracy remained the same which means that system did overfit to some extent as it performed well on training data but not on testing data.
         4. Support Vectors:

There are 63 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.14377853 0.11001499 0.75 4.46599757]

* 1. Implemented a Gaussian SVM learner
     1. The classification accuracy with various C and gamma values are as follows
        1. C=1, gamma=1
           1. Train\_accuracy=0.69
           2. Test\_accuracy=0.66
           3. Here both train and test accuracy are close enough and similar to the linear SVM classification accuracy.
           4. Support Vectors:

There are 69 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.0645591 0.03 0.12706005 2.02979085]

* + - 1. C=2, gamma=1
         1. Train\_accuracy=0.76
         2. Test\_accuracy=0.66
         3. Here both train and test accuracy are close enough and similar to the linear SVM classification accuracy. Also the testing accuracy didn’t change even If the training accuracy was increased which suggests overfitting.
         4. Support Vectors:

There are 69 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.0645591 0.03 0.12706005 2.02979085]

* + - 1. C=5, gamma=1
         1. Train\_accuracy=0.77
         2. Test\_accuracy=0.72
         3. Here both train and test accuracy are close enough and similar to the linear SVM classification accuracy. Also the testing accuracy increased as per the training accuracy.
         4. Support Vectors:

There are 67 support vectors for this which were close to the decision boundary.

Examples:

[0.13819189 0.11032805 0.75 2.87671687]

[0.0645591 0.03 0.12706005 2.02979085]

* + - 1. C=6, gamma=5
         1. Train\_accuracy=0.91
         2. Test\_accuracy=0.88
         3. Here the accuracy for training and test increased by a substantial amount.
         4. Support Vectors:

There are 66 support vectors for this which were close to the decision boundary.

Examples:

[0.094423 0.11009195 0.6661024 4.13041801]

[0.0645591 0.03 0.12706005 2.02979085]

1. Implementing Decision tree to classify three classes “Metal”,”Plastic”,”Ceramic”.
   1. The write up is present in 3a.txt.
   2. The decision tree is implemented where training accuracy is computed for various depth and no testing data was used for this section.
      1. Depths
         1. For depth=1 we get an accuracy of 61% and for depth=2 it increased to 76%.
         2. For depth=5 and depth=6 it remained same also for depth=10 and depth=11 the accuracy remained same at about 99% which shows that the tree is overfitting.
   3. Here as we have a set of training and testing data, we compared classification accuracy for various depths.
      1. The testing accuracy increased from depth=1 at 61% till depth=8 at 72%.
      2. The training accuracy increased from depth=7 to 8 but testing accuracy remained same at 72% which shows a probable overfitting of the decision tree.