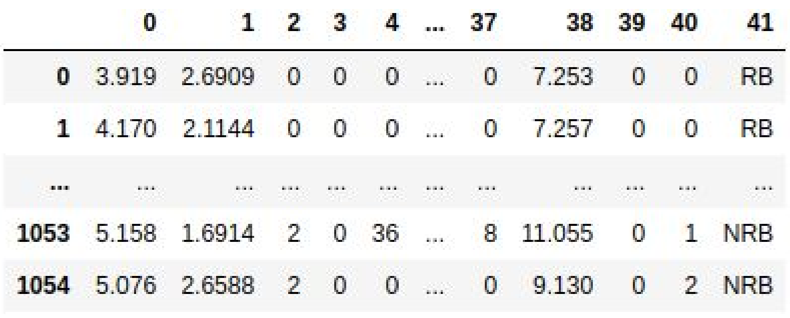
Assignment 3
SVM & ANN

Group-8 -> Aryaman Jain (18CS30007) | Archit Agarwal(18CS10006)

Objective:

Given Dataset:



	0	1	2	3	4	...	37	38	39	40	41
0	3.919	2.6909	0	0	0	...	0	7.253	0	0	RB
1	4.170	2.1144	0	0	0	...	0	7.257	0	0	RB
...	...	...	...	...	...	...	...	...	...	...	...
1053	5.158	1.6914	2	0	36	...	8	11.055	0	1	NRB
1054	5.076	2.6588	2	0	0	...	0	9.130	0	2	NRB

(Table shown above contains some features only, whereas total features are 42)

- It contains the record of 1055 chemicals, along with their 41 molecular descriptor values and 1 experimental class defining their biodegradability.

- Hence, there are 42 features provided in total including target feature, which are mentioned here.
  Target feature is the biodegradation experimental class of the chemicals(Column 41) .

- We have to classify the chemicals on the basis of their biodegradability, i.e. decide whether their experimental class label should be Readily Biodegradable(RB) or Not Readily Biodegradable(NRB).

Preprocessing:

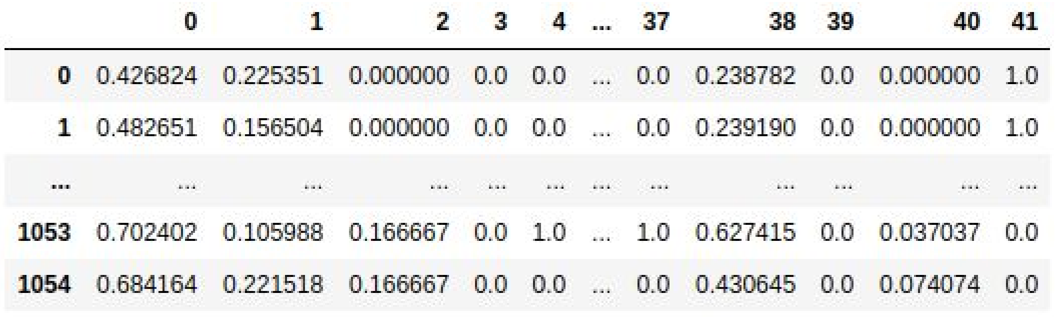
- The given data regarding molecular descriptor values of chemicals was Normalised, and hence was scaled down to the range [0,1], using the formulae:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

- The target feature, which had two classifications were converted to integers as:

RB	1.0
NRB	0.0

Preprocessed Dataset:



	0	1	2	3	4	...	37	38	39	40	41
0	0.426824	0.225351	0.000000	0.0	0.0	...	0.0	0.238782	0.0	0.000000	1.0
1	0.482651	0.156504	0.000000	0.0	0.0	...	0.0	0.239190	0.0	0.000000	1.0
...	...	...	...	...	...	...	...	...	...	...	...
1053	0.702402	0.105988	0.166667	0.0	1.0	...	1.0	0.627415	0.0	0.037037	0.0
1054	0.684164	0.221518	0.166667	0.0	0.0	...	0.0	0.430645	0.0	0.074074	0.0

Implementation:

We read the data set(.csv file) using pandas library.

Part 1

- We randomly split the data into 80:20 where 80 parts are for training and the remaining 20 parts are for testing.
- We used sklearn svm library to implement support vector machines
- The values of C taken were 0.001,0.01,0.1,1,10,100,1000
- Then we passed all C values for linear , radial basis function and quadratic kernel and simultaneously predicted accuracy over the test set
- Then we find the best accuracy of the above models among all C values.

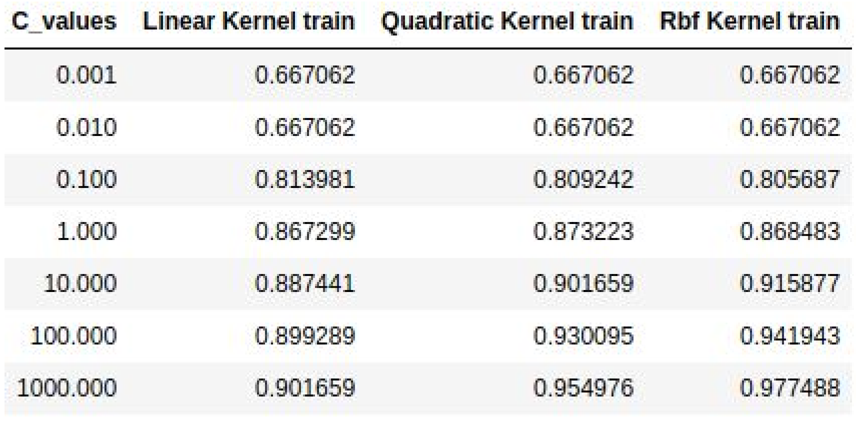
Part 2

- We randomly split the data into 80:20 where 80 parts are for training and the remaining 20 parts are for testing.
- We then build a class *MyDataset* which returns length of dataset and features values,target value of the specified index in case of training data and only features values in case of test data
- Then we made an instance of *MyDataset* class and feed the output to Data Loader which slices the train data and test data into batches of specified size
- We made a class *MyModel* which takes input a list that contains the number of nodes in a hidden layer and out size which is set to 1 since there are only two classes to predict.
- *MyModel* class outputs a neural network as specified in the hidden layer list with relu as activation function between hidden layers and sigmoid as activation function of the output layer.
- We made *training* function which takes input as training data , hidden layers and learning rate, it used SGD(Stochastic Gradient Descent) optimizer provided by pytorch with the specified learning rate, it outputs the neural network model with the updated weights after backpropagation
- *testing* function takes input a model and test data , it feeds test data into the model and outputs a numpy array of predicted classes
- *get_acc* function takes input predicted class and test class and return the fraction of examples which are correctly predicted by the model
- *layers* list contains the all hidden layers with first input as input layer (number of features) and learning rate contains all learning_rate specified in the question
- We predicted test_data for each model with varying_learning rates and plot the graph of accuracy vs learning_rates for all 5 models and accuracy vs model for all 5 learning rates

Results:

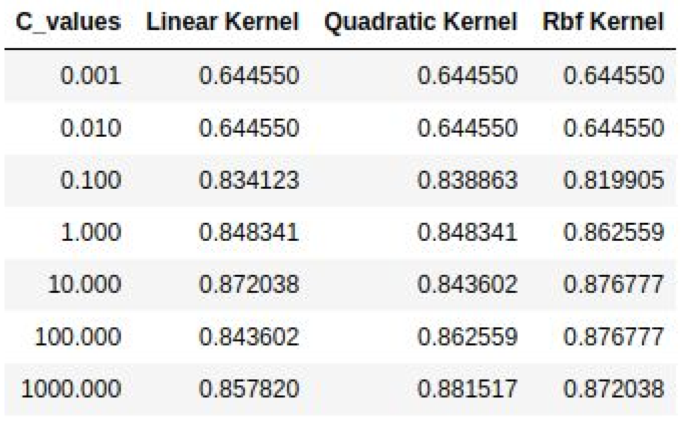
Part 1

Training Set Accuracy:



C_values	Linear Kernel train	Quadratic Kernel train	Rbf Kernel train
0.001	0.667062	0.667062	0.667062
0.010	0.667062	0.667062	0.667062
0.100	0.813981	0.809242	0.805687
1.000	0.867299	0.873223	0.868483
10.000	0.887441	0.901659	0.915877
100.000	0.899289	0.930095	0.941943
1000.000	0.901659	0.954976	0.977488

Test Set Accuracy:



C_values	Linear Kernel	Quadratic Kernel	Rbf Kernel
0.001	0.644550	0.644550	0.644550
0.010	0.644550	0.644550	0.644550
0.100	0.834123	0.838863	0.819905
1.000	0.848341	0.848341	0.862559
10.000	0.872038	0.843602	0.876777
100.000	0.843602	0.862559	0.876777
1000.000	0.857820	0.881517	0.872038

Most Suitable C values(corresponding to highest test set accuracy):

Linear Kernel	Quadratic Kernel	Rbf Kernel
10.000	1000.000	100.000

Test and Training Accuracy corresponding to most suitable C value:

Kernel	Test Accuracy	Training Accuracy
Linear Kernel	0.872038	0.887441
Quadratic Kernel	0.881517	0.954976
Rbf Kernel	0.876777	0.941943

Note:
Test accuracy for Rbf kernel is same for C = 10.000 and C = 100.000
So, most suitable value of C is chosen as C = 100.000, as it had better training accuracy.

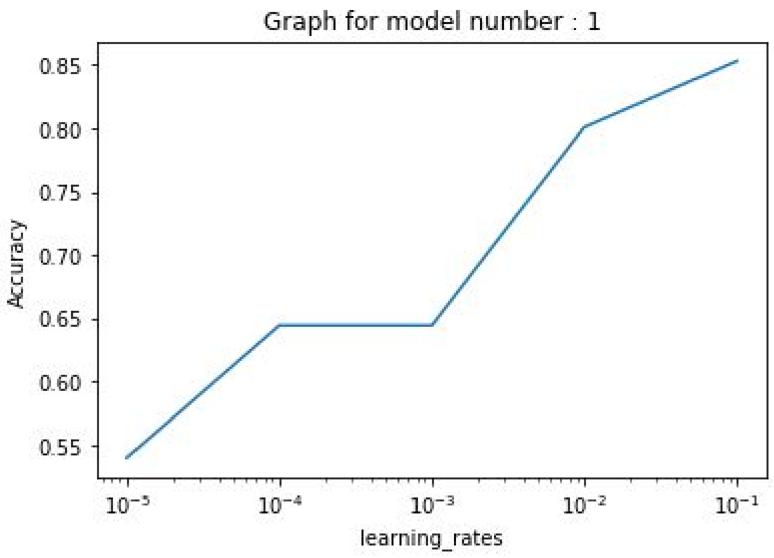
Part 2

Number of nodes in the input layer = 41 nodes
Number of nodes in output layer = 1 node

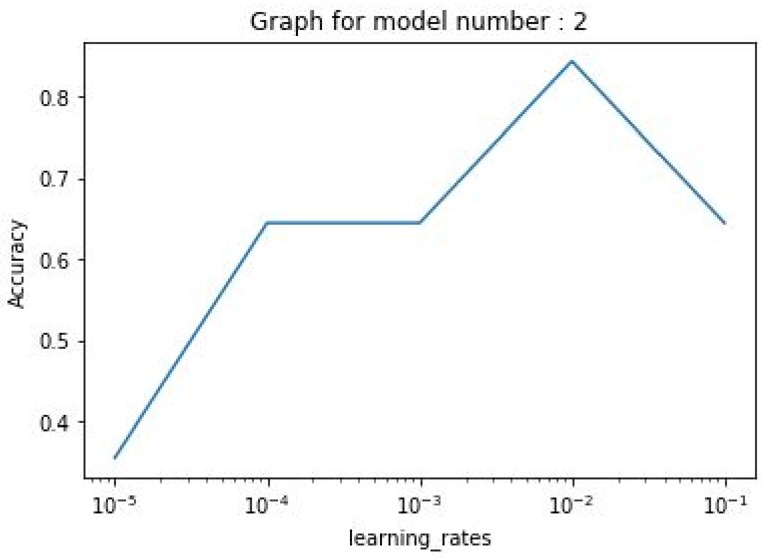
Justification:
Input layer contains 41 nodes as there are 41 features (excluding target feature).
Output layer contains 1 node because there are two classes and we used sigmoid activation layer for the output layer if output is >0.5 then predicted class is 'RB' if output is <= 0.5 predicted class is 'NRB'.

b) Learning Rate vs Accuracy(for each model):

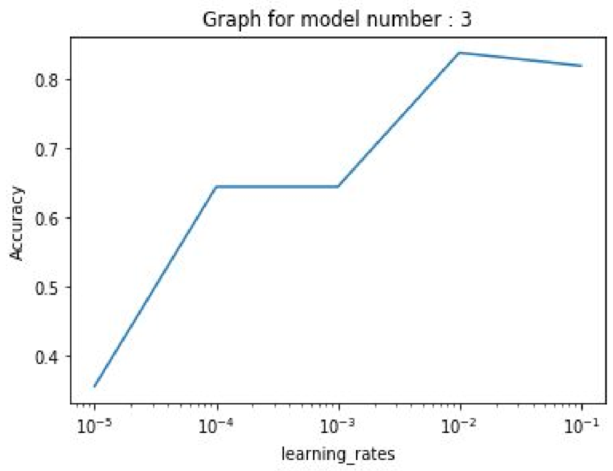
Model 1: 0 hidden layer



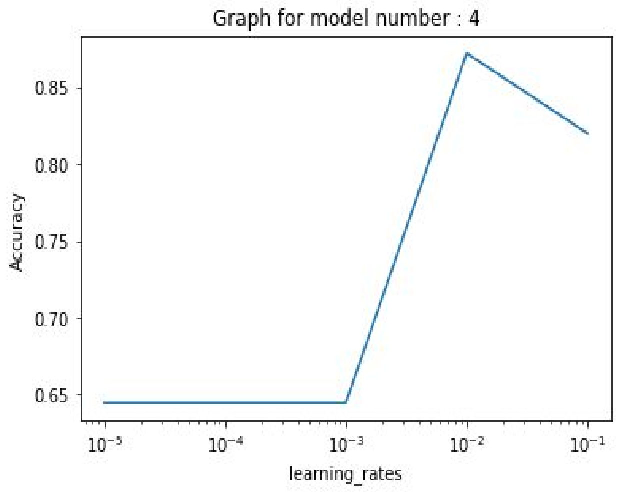
Model 2: 1 hidden layer with 2 nodes



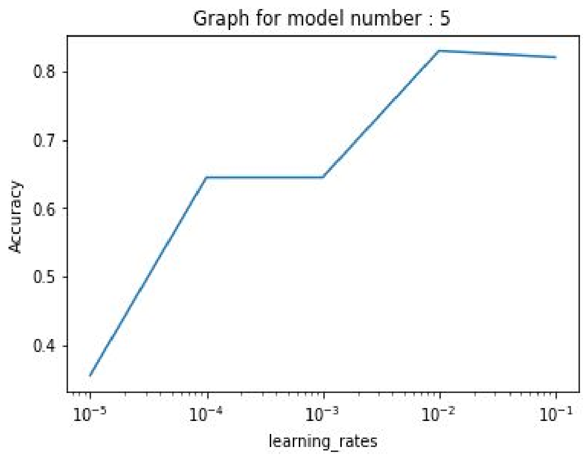
Model 3: 1 hidden layer with 6 nodes



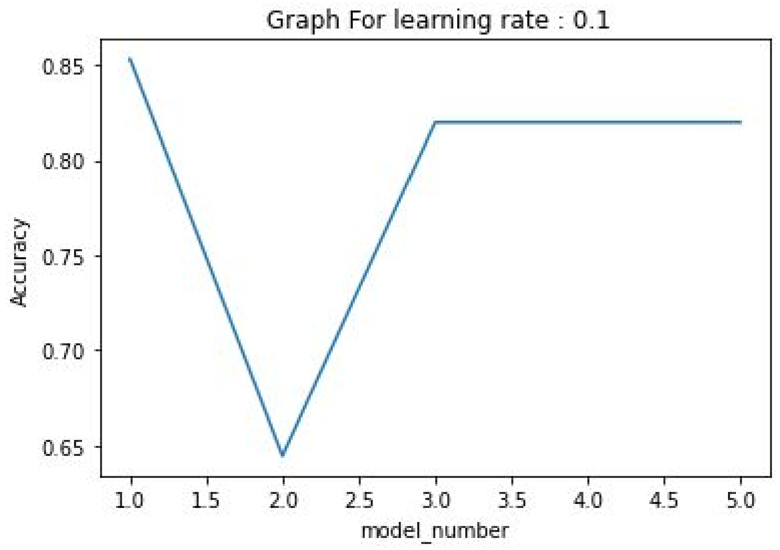
Model 4: 2 hidden layers with 2 and 3 nodes respectively

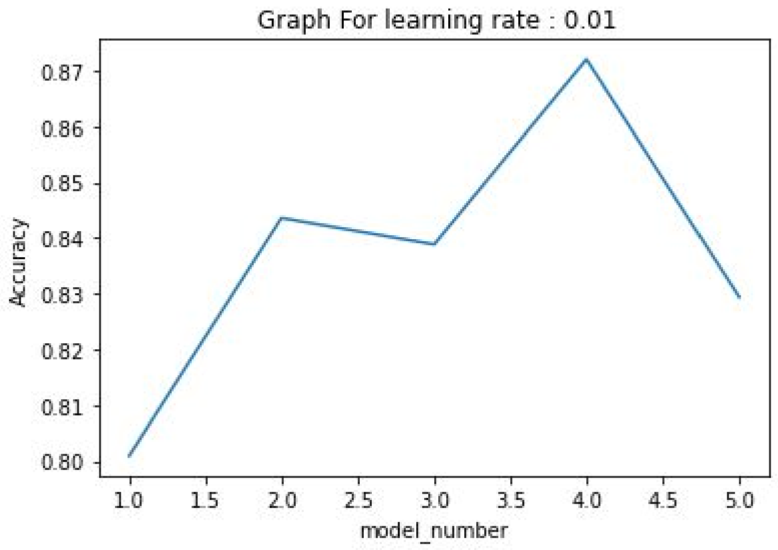


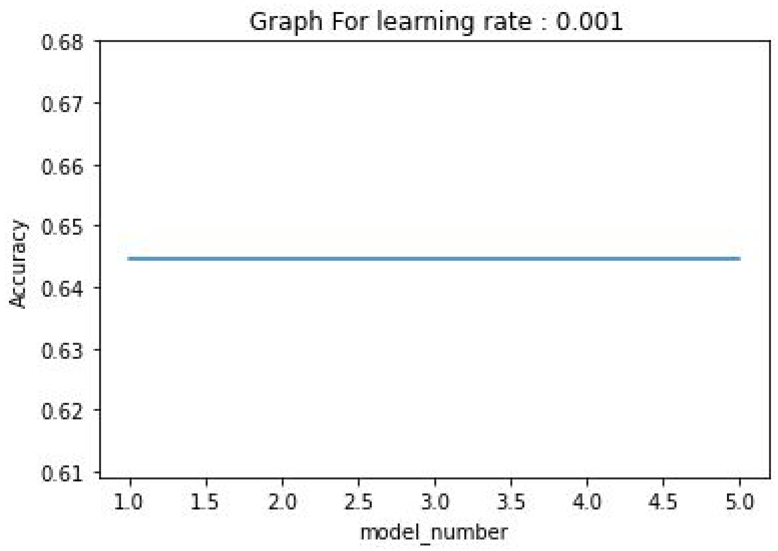
Model 5: 2 hidden layers with 3 and 2 nodes respectively

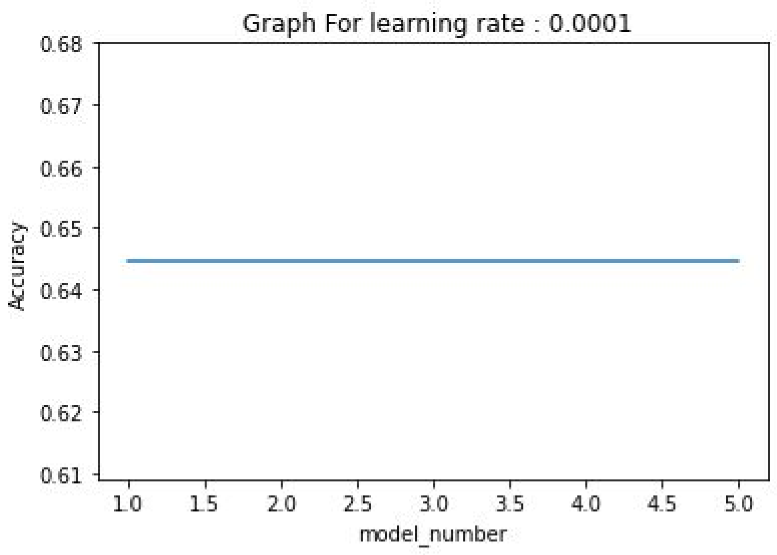


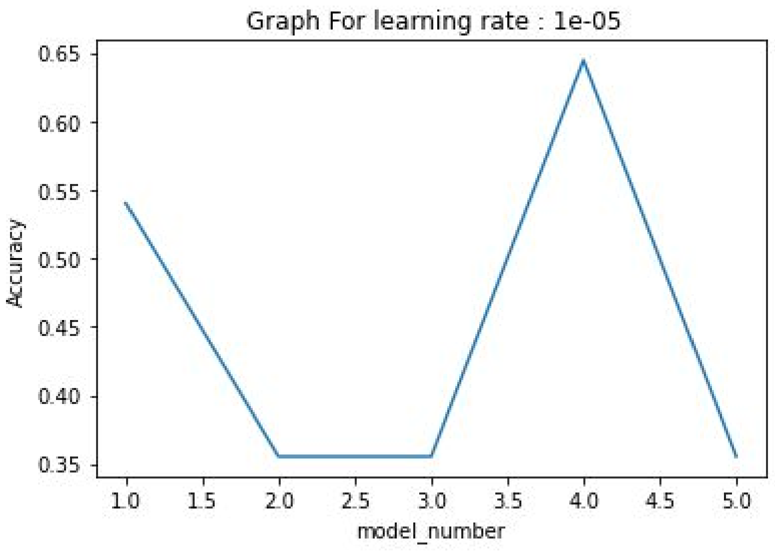
Models vs Accuracy graphs(For each learning rate):











c) The hyperparameters for the best found model are:

- Model Number : 4 (2 hidden layers with 2 and 3 nodes respectively)
- Batch Size : 32
- Optimizer : Stochastic Gradient Descent
- Learning Rate : 0.01
- Number of epochs : 250
- Weight Decay : 0

Justification:

- Model number 4 with learning rate 0.01 is best because when learning rates are <0.01 there are not enough epochs to reach the global minimum of the loss function and when learning rate is greater than 0.01 the loss is not as convergent as in case of 0.01. Model 4 has more parameters (connections) than model 1 and model 2 so it is able to fit more accurately the training data but model 3 and model 5 has more number of parameters than model 4 and in that case model 3 and model 5 tends to overfit the training data hence less training accuracy than model 4.

- We used batch size as 32 because of the fact that SGD does not perform well when batch is large also number of epochs are set to be 250 because if we set epochs to be more than 250 considerable change in training accuracy cannot be seen in case of learning rate of 0.1 and 0.01 and for other learning rates we would require epochs in range of ~10000 which would increase training time by hours unnecessarily and there is also a high probability of overfitting

Part 3

Performance Comparison:

Support vector machines seem to perform better than artificial neural networks for this problem. SVM gives max accuracy of 88.151 , Artificial Neural Network gives max accuracy of 87.2038. Although the difference in test accuracy is small but SVM gives results in order of milliseconds whereas Artificial Neural Net takes about average of 8-10 sec for each model.

Note:

Running Time of the notebook is around 5-10 min depending on the CPU of the system.