CSL2050 PATTERN RECOGNITION AND MACHINE LEARNING

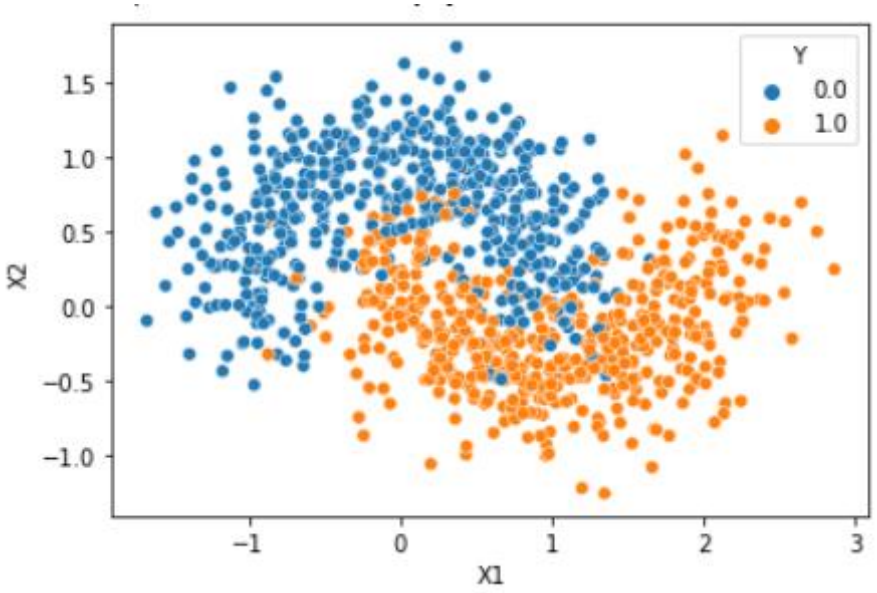
LAB REPORT-05

NAME:	SAMEER SHARMA
ROLL NUMBER:	B21CS066
LAB TITLE:	Bagging and Boosting

Q1)

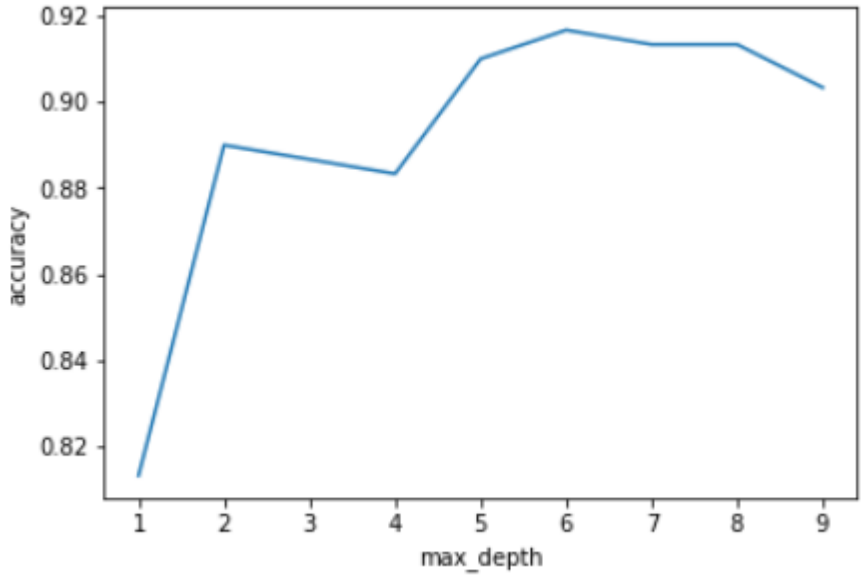
Part 1)

- Using the make moons function of sklearn, created a synthetic dataset with noise of 0.3. Also, assigned the random state parameter equal to 42 so that we get similar dataset every time we run the cell.
- After this reduced the noise of dataset by using the standard scaler function from sklearn. The obtained dataset is as follows.



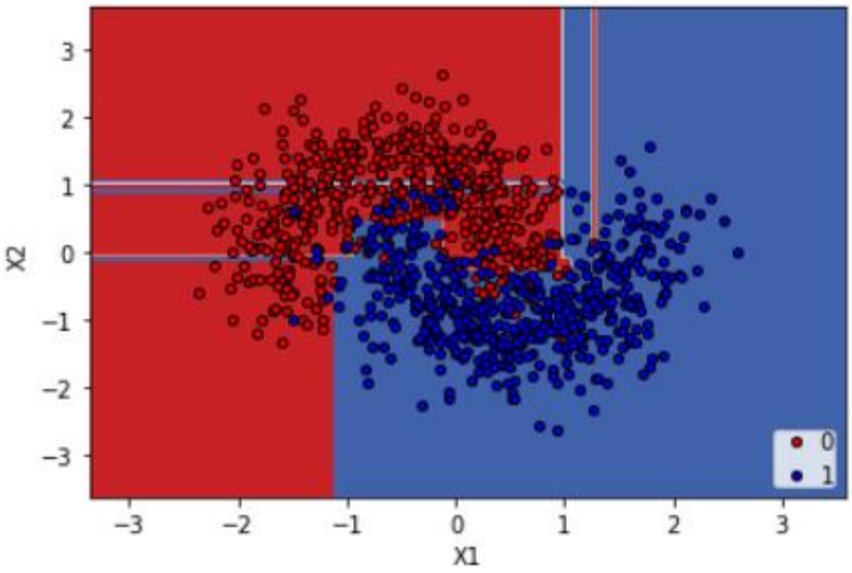
- In the last splitted the dataset into training and testing dataset using train test split function from sklearn with test_size=0.3.

- For hyper parameter tuning of max depth of the decision tree, I iterated over a range from 1 to 10 and plotted the accuracies on different values of max depth. Got the maximum accuracy of around 91.6% at max depth = 6. Plot for the same is as follows

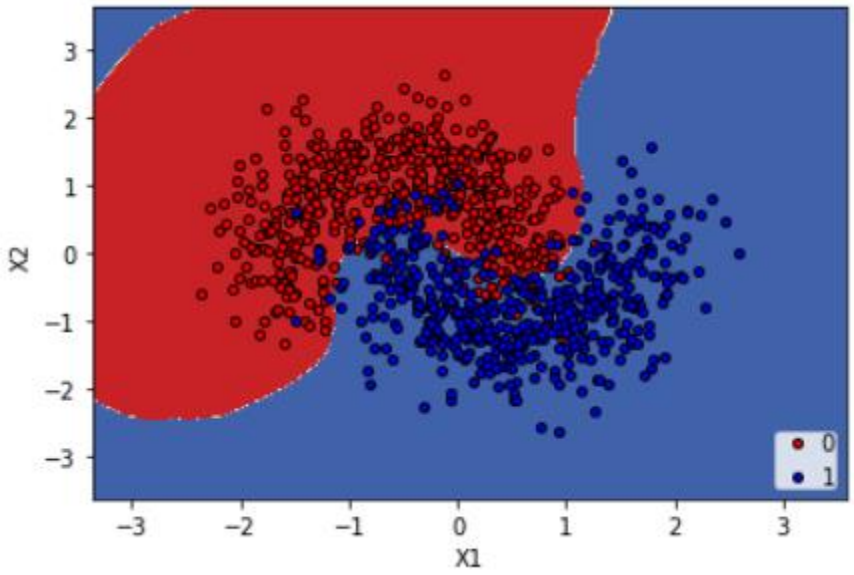



```
0.9166666666666666 6
```

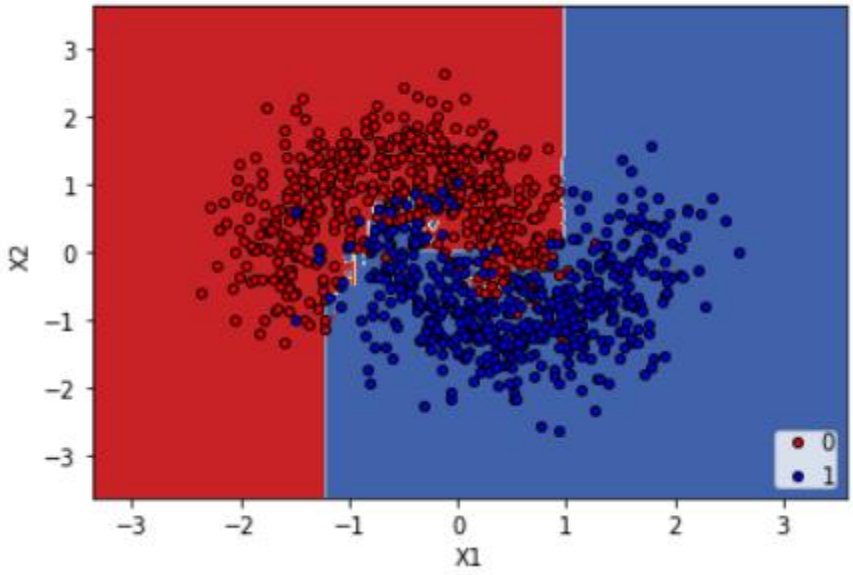
- The obtained decision boundary of the decision tree at max depth = 6 and trained over training data is as follows



- Trained a Bagging classifier from sklearn library and trained it over the training data. The obtained decision boundary is as follows.

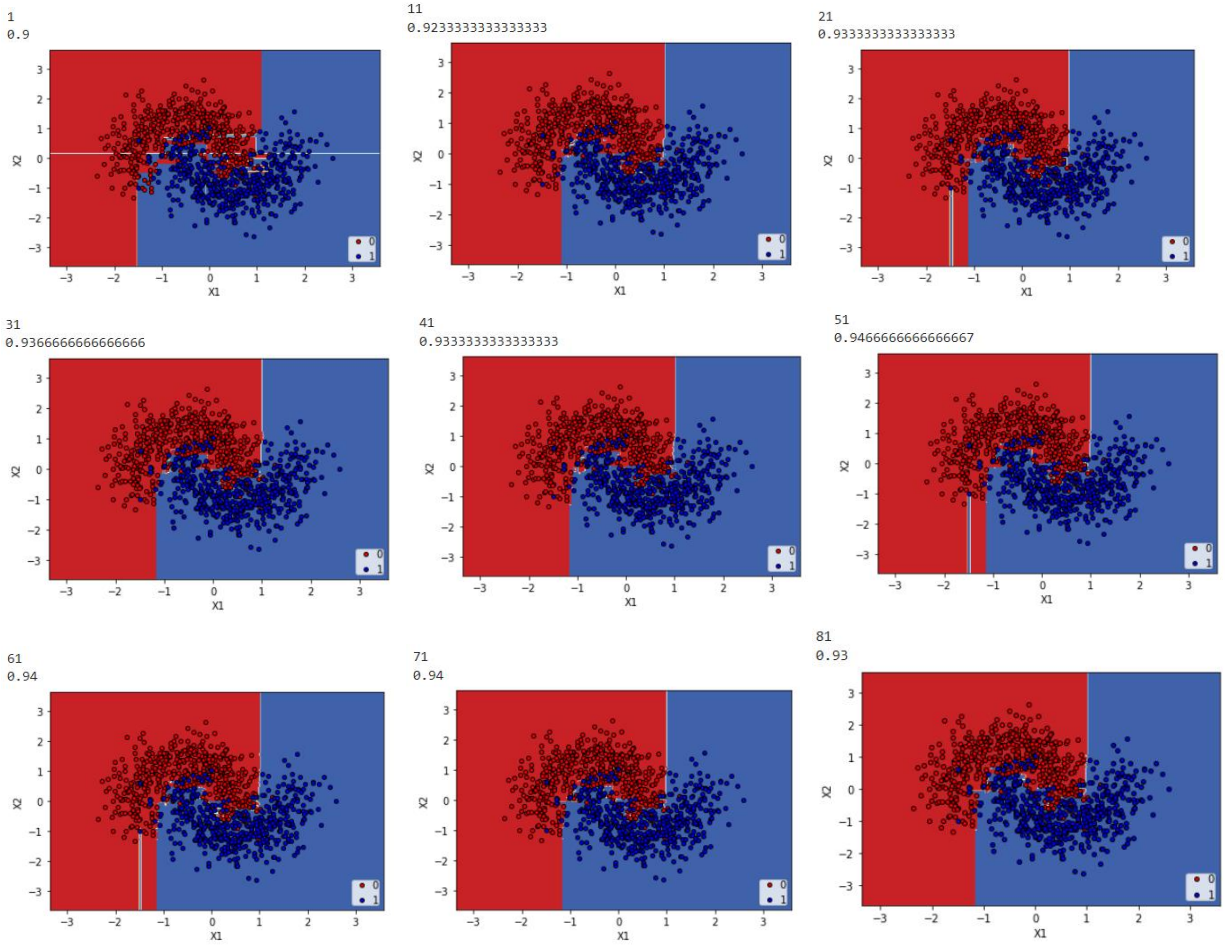


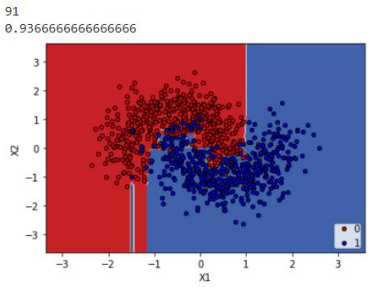
- Similarly for Random forest classifier



- The accuracies for the three classifiers on testing data are as follows
  - Decision tree classifier -> 0.9166666666666666
  - Bagging classifier -> 0.9233333333333333
  - Random Forest classifier -> 0.9366666666666666

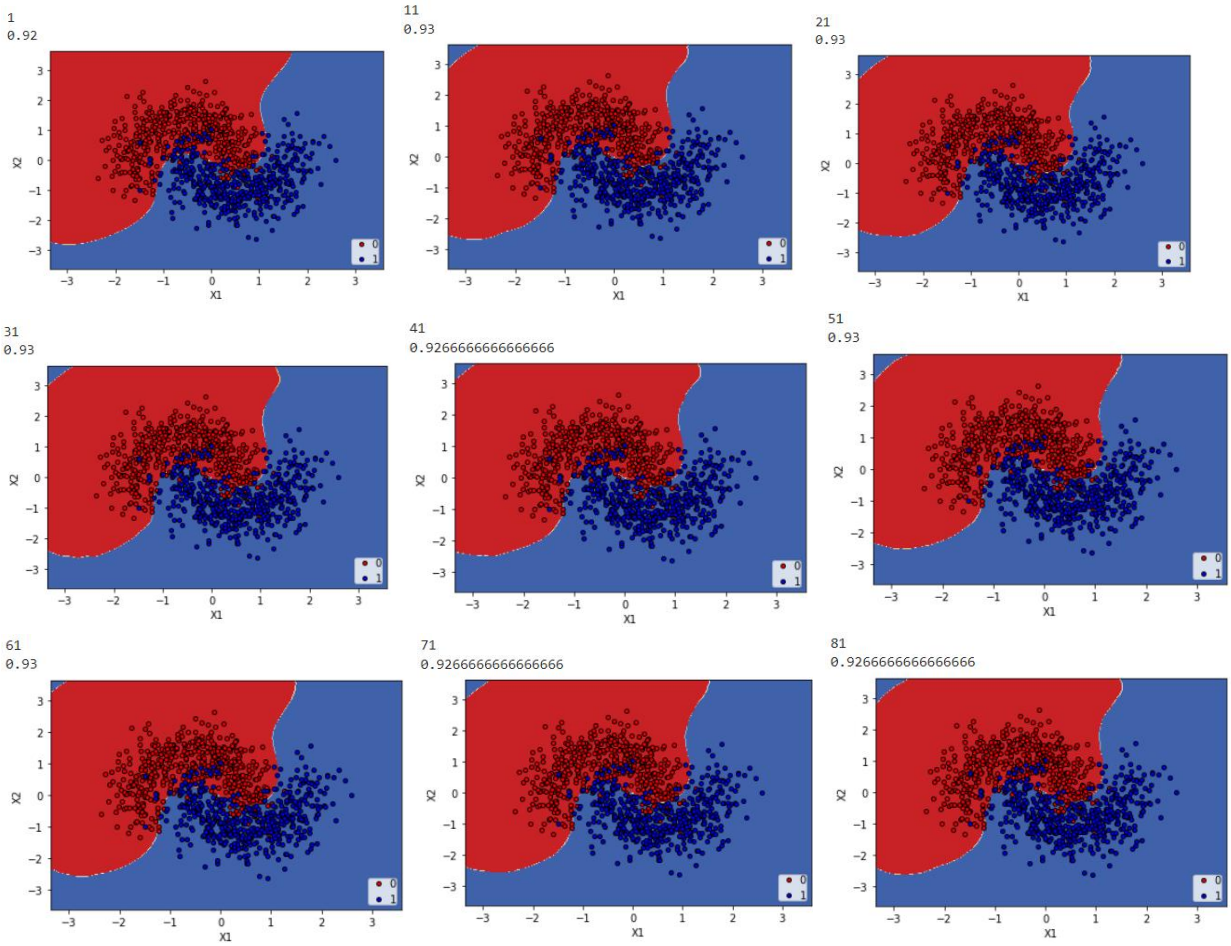
- Clearly, the accuracies of the three models are not varying much. Still, accuracy of random forest classifier is highest and that of decision tree classifier is least.
- The decision boundaries for Decision tree classifier and random forest classifier consists of straight lines and thus form staircase type structure while that for bagging classifier consists of curves.
- Now, let us vary the number of estimators and see its effect on the accuracy and decision boundary of the Bagging classifier and Random forest classifier.
- For Random forest classifier



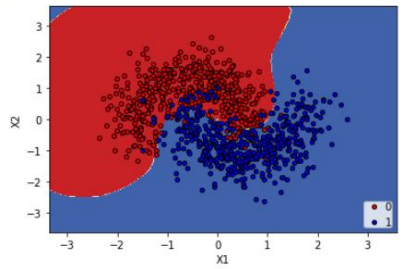


- The number of estimators in the classifier and their accuracy are on the top of each graph. We can see that the accuracy of random forest classifier is increasing a little bit on increasing the number of estimators. Here, the accuracy for each classifier is calculated on testing data.

- For Bagging classifier



91
0.93



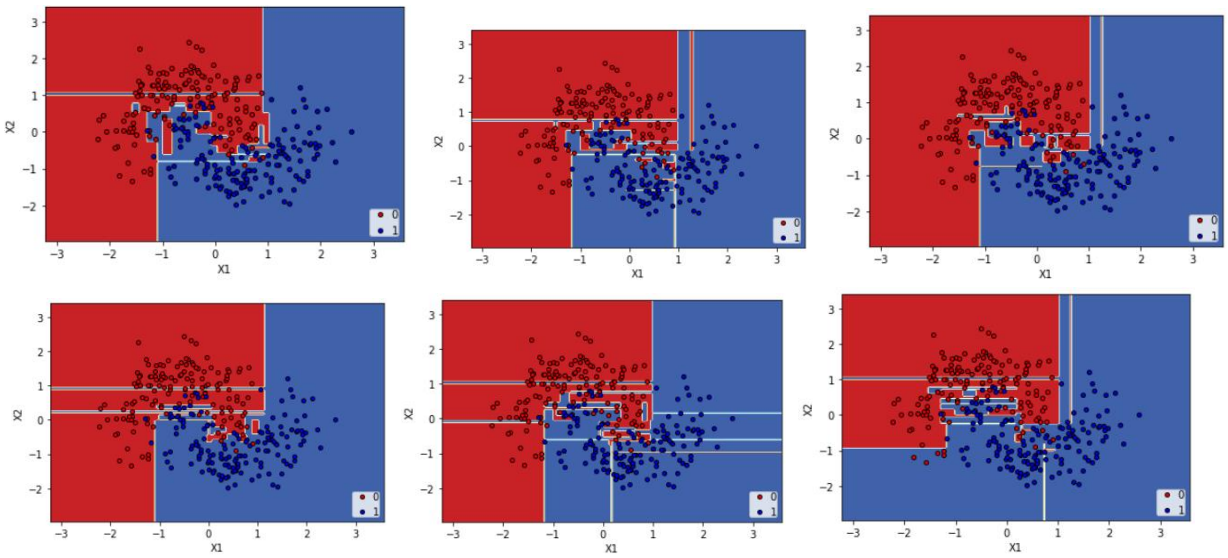
- The accuracy of every classifier is nearly around 93%.

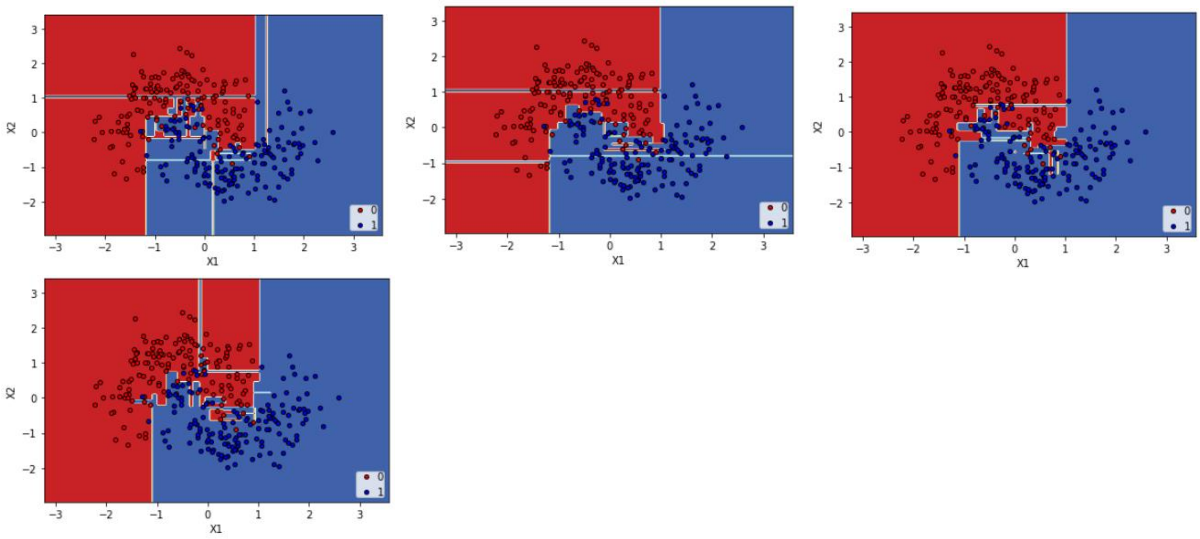
Part 2)

- Programmed a Bagging classifier from scratch whose well commented code is in the google colab file.
- Then, trained a Bagging classifier on the previous data set and with 10 estimators in the ensemble and checked its accuracy on the testing data. Got accuracy of `0.93` on the testing dataset.
- Accuracy of each individual estimator and their mean are as follows

```
[0.9133333333333333, 0.8766666666666667, 0.9, 0.8966666666666666, 0.8766666666666667, 0.8566666666666667, 0.91, 0.8933333333333333, 0.9033333333333333, 0.91]
0.8936666666666667
```

- The accuracies of each individual estimator is in the range 84% to 92% and mean of the accuracies is around 89%.
- Decision boundaries for each individual estimator are as follows





- The decision boundaries are nearly same for each estimator excluding some rectangles intruding in the plot (This is because different dataset is fed to different estimator and these rectangles are result of multiple occurrences of same data point in the training data). These errors gets reduced when we ensemble them and the overall accuracy increases. This we can also confirm from the fact that we build a model with accuracy of around 93% from the classifiers with mean accuracy of around 89%.

Problem 2)

Part 1,2,3)

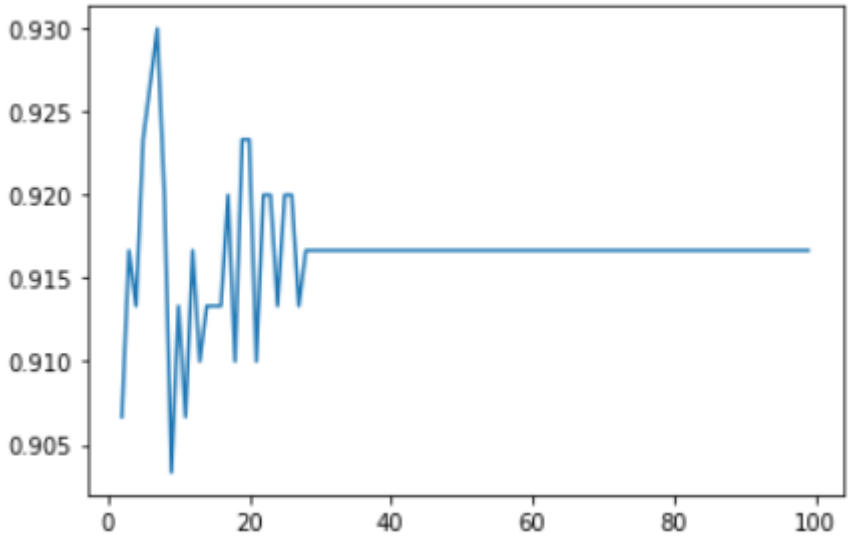
- Trained a Adaboost and a XGBoost model on the previous training dataset. The subsample value is 0.7 for the XGBoost model. Accuracies of these two models on the training data and testing data are as follows

- As expected, the accuracy on the training data is more than that on the testing data.

```
accuracy of Adaboost model on training data 0.9385714285714286
accuracy of Adaboost model on testing data 0.92
accuracy of XGboost model on training data 0.9471428571428572
accuracy of XGboost model on testing data 0.9166666666666666
```

Part 4)

- Next, trained a LightGBM model on the training data for different values of number of leaves in tree and plot of the accuracy vs num_leaves is as follows



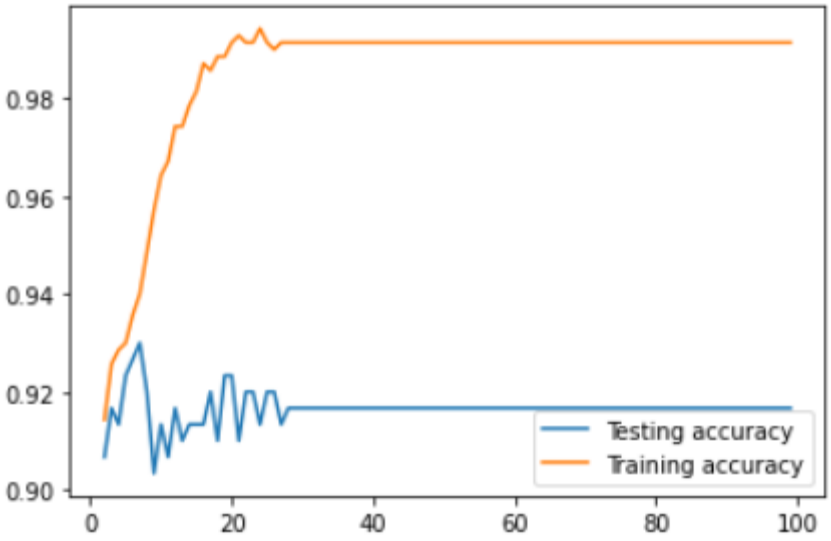
- We are getting maximum accuracy of 93% on num_leaves=7.

Part 5)

- The general relation between num_leaves is max_depth is as follows.

$$num_{leaves} = 2^{\max\_depth}$$

- But it is not always true. Light GBM models grow depth wise and not breadth wise like decision tree classifiers. Generally, for small value of num_leaves, we can have large value of max_depth. In general, they are proportional to each other.
- Plot for accuracy of the lightGBM model on the training data and testing data is as follows



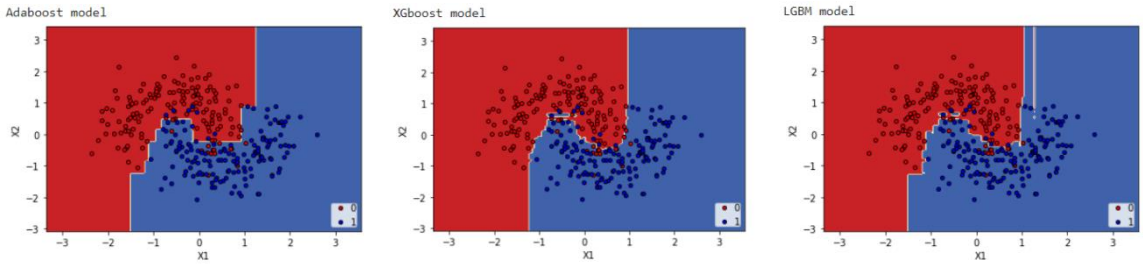
- Clearly, till num_leaves = 7, both training accuracy and testing accuracy are increasing and nearly equal. But after that, testing accuracy starts decreasing but training accuracy is still increasing. So, we can say that around num_leaves = 7, the model starts overfitting the training data.

Part 6)

- Parameters that can be used to increase the accuracy:-
  - Use large max_bin (This can increase time complexity)
  - Use small learning_rate with large num_iterations.
  - Use large num_leaves (but increase till particular extent else it will overfit)
  - Use training data with more datapoints.
- Parameters that can be used to deal with overfitting
  - Decrease num_leaves (to a extent else it will underfit)
  - Alter the value of min_data_in_leaf and min_sum_hessian_in_leaf
  - Use feature sub-sampling similar to what we did in XGBoost model
  - Use bigger training dataset
  - Alter max_depth to avoid overfitting
  - Alter min_gain_to_split to early stop the tree

Part 7)

- Decision boundary for the three models are as follows

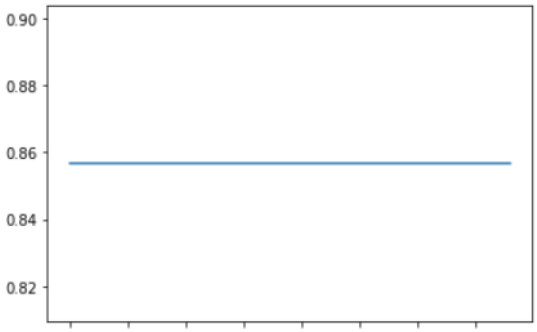


- Performance of the three models on the testing data is as follows

```
accuracy of Adaboost model 0.92
accuracy of XGboost model 0.9166666666666666
accuracy of LGBM model 0.93
```

Q3)

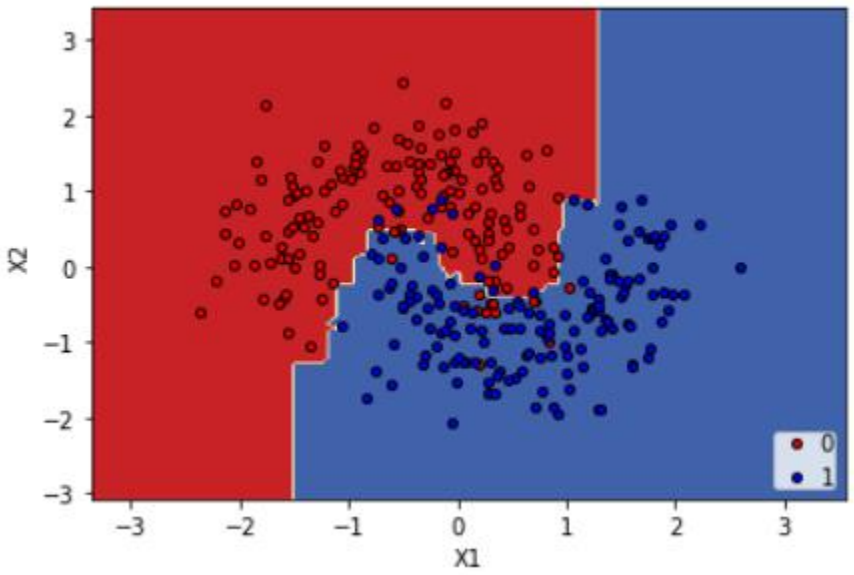
- Here I am choosing Gaussian Bayes classifier since we have continuous features. It has two parameters:-
    - Priors: Sets the prior probabilities for each class.
    - Var_smoothing: Portion of the largest variance of all features that is added to variances
- Clearly, we can only alter var_smoothing for hyper parameter tuning (We value of prior probabilities are [0.5, 0.5] and we can lose accuracy if we alter it).
- For me, accuracy on the testing data is not changing



- For Voting classifier, I chose Adaboost , XGBoost, LightGBM and Bayes model. Set the value of parameter voting to hard since we want majority voting. Then, trained the voting classifier on the training data and its accuracy on the testing dataset is as follows.

```
0.9133333333333333
```

- Obtained decision boundary is as follows



- Comparison between the five classifiers is as follows

```
Accurcay of Bayes model 0.8566666666666667
Accurcay of XGBoost model 0.9166666666666666
Accurcay of Adaboost model 0.92
Accurcay of LightGBM model 0.93
Accurcay of Voting classifier 0.9133333333333333
```

- The mean of first four classifiers is around 90%. Clearly, voting classifier is more accurate than the average performance of other four classifiers.