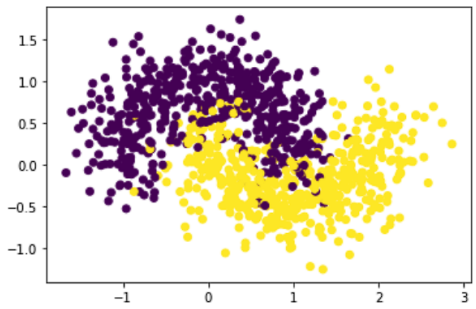
Lab Assignment 5
Prakhar Gupta
B21AI027

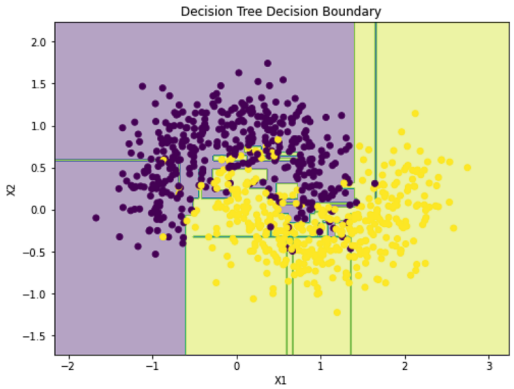
Question 1:

Part 1-

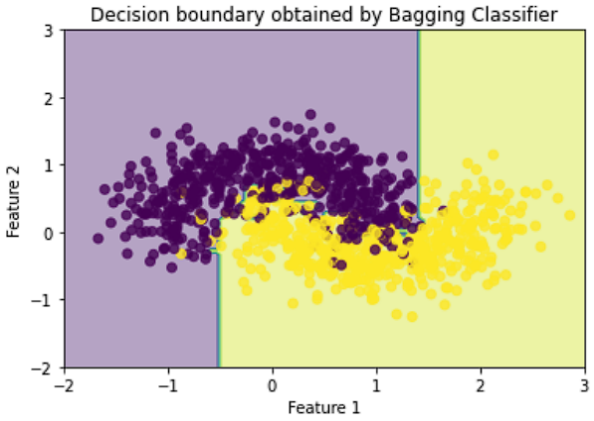
- Made **X,y** using **sklearn make_moons** function(random_state=42, noise=0.3,n_samples=1000)
- Converted **X,y to df** and then used **df.describe()**
- Checked **df.isnull().sum()**
- Plotted X,y



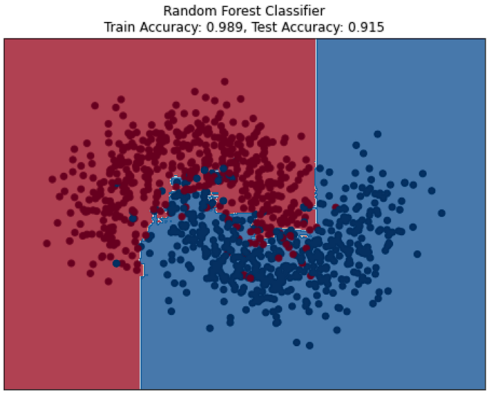
- Used train_test_split to split train:test in **80:20**
- Fitted Decision Tree Classifier



- Plotted DT boundary
- Finded best params for DT using **GridSearchCV**
- Trained a Bagging Classifier
- Plotted Decision boundary



- Trained a RandomForestClassifier
- Plotted Decision Boundary



- Best accuracy order of Train_data - **Random Forest > Bagging Classifier > Decision Tree**
- Best accuracy order of Test_data - **Random Forest > Bagging Classifier > Decision Tree**
- We can see Decision Tree has **high bias**, Bagging Tree is somewhat **overfitting** and **Random Forest is the best**
- Varied the number of estimators for the BaggingClassifier and RandomForestClassifier and plotted Decision Boundaries
- From The Accuracy we can clearly see that it sometimes up and down but on long run it clearly increases
- The best accuracy we get are:
- bst_train_randforest_accuracy: 0.99375
- bst_test_randforest_accuracy: 0.915
- bst_train_bag_tree_accuracy: 0.99375
- bst_test_bag_tree_accuracy: 0.91
- We can see the randomforest and bag_tree accuracy get **equal at high estimators** value in train set
- We can see the randomforest accuracy is greater than bag_tree accuracy at high estimators value in test set

- From plots of both bag_tree and random forest we can see that the boundary becomes more and more **complex** as n_estimators value increases

Part 2-

- Implemented Bagging algorithm from scratch
- Applied scratch bagging algorithm with **n_estimators = 10**
- Trained the Bagging algo got its accuracy `0.92`
- Average accuracy on trees is `0.9`
- The average accuracy of all the trees is 0.9 whereas for bagging it is 0.92
- Bagging accuracy > Average accuracy of all trees

Question 2:

Part 1-

- Trained AdaBoostClassifier with n_estimators=100

Part 2-

- Trained a XGBClassifie with n_estimators=100, and subsample=0.7

Part 3-

- Printed accuracy of both AdaBoostClassifier and XGBClassifie on train and test data

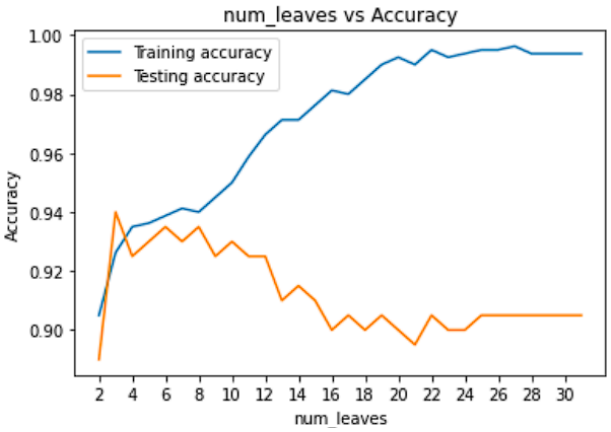
```
AdaBoost Classifier Accuracy on training set:  0.9475
AdaBoost Classifier Accuracy on test set:  0.91
XGBoost Classifier Accuracy on training set:  0.9925
XGBoost Classifier Accuracy on test set:  0.905
```

Part 4-

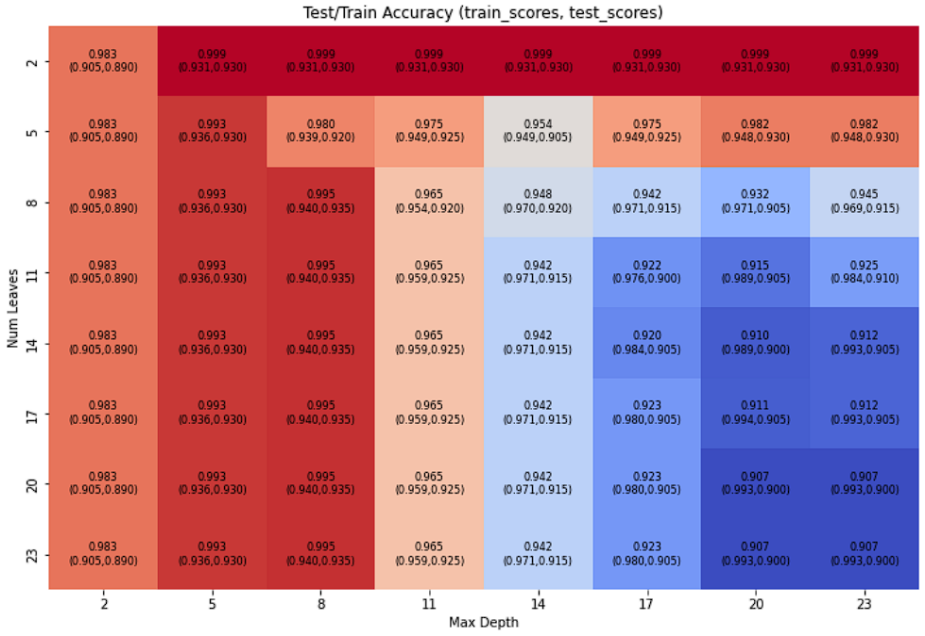
- Trained a LightGBM model
- Varied different values for num_leaves

Part 5-

- Analysed relation between max_depth when num_leaves is given to the LightGBM model
- Plotted accuracy for various num_leaves value



- Plotted heatmap where axis are max_depth and num_leaves and the y value is test/train accuracy



- From num_leaves vs Accuracy plot we can see the model starts overfitting after **num_leaves=8 and max_depth=8**
- From heatmap also we can see the same result

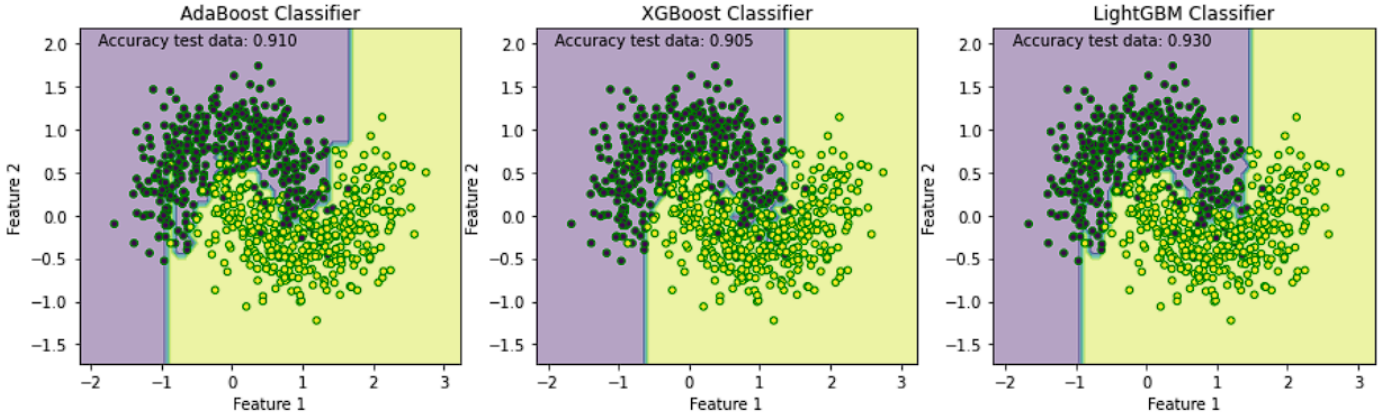
Part 6-

- **num_leaves**: This decides how many leaves will be in a tree. If we increase num_leaves, it can make the prediction better.
- **max_depth**: High depth can also lead to overfitting.But it will increase the accuracy
- **min_data_in_leaf**: This parameter sets the minimum number of samples required to be at a leaf node. If we increase this value, we will have less overfitting for each leaf, which can help to prevent overfitting.

- **feature_fraction**: This decides how many features we want to use in each tree. If we lower this value, we will have less overfitting, which can help us to prevent overfitting.
- **lambda_l1 and lambda_l2**: These parameters decide L1 and L2 regularization, respectively. If we set non-zero values for lambda_l1 and lambda_l2, it can help us to prevent overfitting.

Part 7-

- Plot the decision boundaries for all the 3 models and compared their performance



- AdaBoost Classifier Accuracy on training set: 0.9475 and on test set: 0.91
- XGBoost Classifier Accuracy on training set: 0.9925 and on test set: 0.905
- LightGBM Classifier Accuracy on training set: 0.9375 and on test set: 0.93
- We can clearly see that accuracy/performance is **LightGBM>AdaBoost>XGBoost**

Question 3:

Part 1-

- Train a Bayes classification model and also tuned the hyperparameter(i.e. var_smoothing)

```
Best parameters found: {'var_smoothing': 1e-19}
Gaussian Naive Bayes Classifier Accuracy on training set:  0.86
Gaussian Naive Bayes Classifier Accuracy on test set:  0.82
```

- The best models till now are RandomForestClassifier, LightGBM Classifier, AdaBoost Classifier

- Made a VotingClassifer from sklearn using (Bayes Classification model,RandomForestClassifier, LightGBM Classifier, AdaBoost Classifier)
- Trained VotingClassifier
- ```
  Voting Classifier Accuracy on training set:  0.95125
  Voting Classifier Accuracy on test set:  0.915
  ```
- Voting Classifier Accuracy on training set: 0.9475 and on test set: 0.91
- Average Accuracy on training set by AdaBoost, Randomforst, LightGBM: 0.95833 and and on testing set: 0.91833
- AdaBoost Classifier Accuracy on training set: 0.9475 and on test set: 0.91
- Random Forest Accuracy on training set: 0.95125 and on test set: 0.915
- LightGBM Classifier Accuracy on training set: 0.9375 and on test set: 0.93
- In comparison to the three model Voting Classifier has both train_accuracy and test_accuracy good
  ```