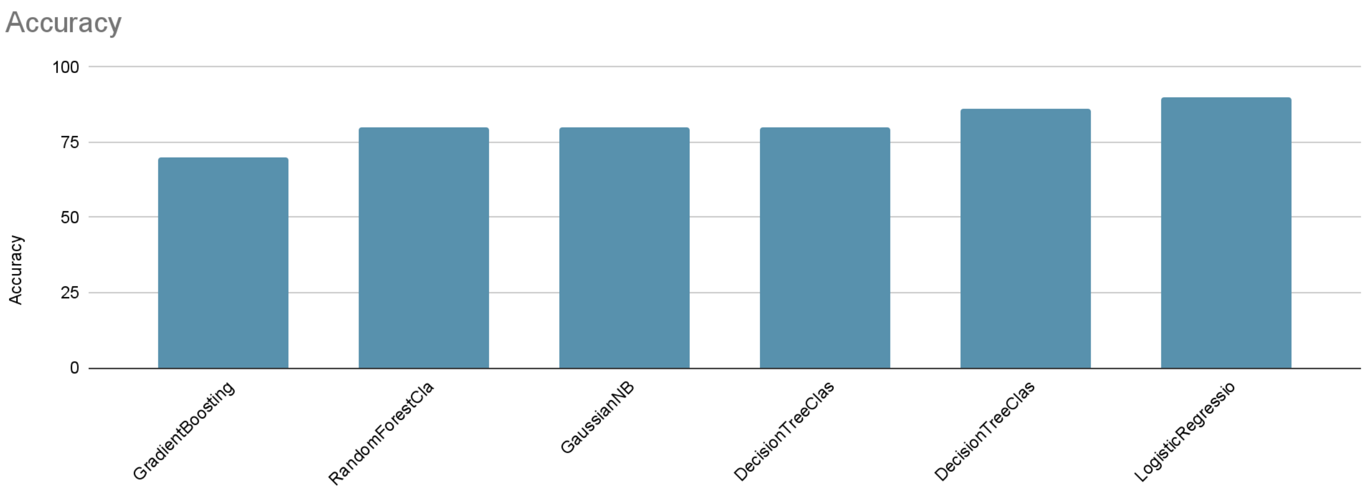
Deepam Sarmah
2020050
deepam20050@iiitd.ac.in

- **Observations**:
  - I observed that the target labels either belong to class ‘0’ or class ‘1’. This seemed to be a classification problem. Also, as there were only two classes I observed that this would be a binary classification problem.
  - I also noticed that the dataset wasn’t that big and neither were there many features assigned to each sample so there wasn’t an issue with time complexity of binary classification algorithms.
- **Methodologies**:
  - As it was my first time using only the sk-learn library for ML I read up the documentation of sk-learn’s different binary classification algorithms. These included GradientBoostingClassifier, RandomForestClassifier, GaussianNB (Naive Bayes Classifier), LinearDiscriminant Analysis, QuadraticDiscriminantAnalysis etc.
  - For each such classifier I wrote code to first divide the training dataset into a training set and a validation set.
  - Next I used the training features and training labels to fit my model using any of the classifiers mentioned above.
  - Post that I used the predict function of those classifiers to predict the outcome on the test set.
  - I then created a pandas dataframe to save my output of Id and target which consisted of Ids of the test data and the predicted labels of the test set using the classifiers.
  - I then saved the output into a csv file for submission into Kaggle.
  - Finally, I used the joblib library to save the models that I created into a .pkl file.
- **Experiments**:
  - I initially tried out GradientBoostingClassifier by altering the n_estimators and min_samples but the accuracy was only 76%. I also changed the parameter called loss and changed the loss function to loss="exponential” but this also didn’t improve the accuracy from 76%.
  - I then tried RandomForestClassifier but it gave only 80% in accuracy metric.
  - I tried the Naive Bayes Classifier which was taught in the first class for binary classification. But even this didn’t give any fruitful results and was only 80% in accuracy metric.
  - I then moved to using DecisionTreeClassifier but this also didn’t give me good results and resulted in only 0.8 in accuracy.
  - I then tried the DecisionTreeClassifier along with the BaggingClassifier. I tweaked the parameters such that n_estimators=1000 and max_samples=150. As I had already observed that the size of the dataset isn’t huge, time complexity wouldn’t be much of an issue in this case. Doing this resulted in 86% accuracy.

- I finally then tried the LogisticRegression algorithm and attained an accuracy of 90% which is the highest.

- **Results**: (Accuracy)
    - GradientBoostingClassifier : 76%
    - RandomForestClassifier : 80%
    - GaussianNB (Naive Bayes): 80%
    - DecisionTreeClassifier: 80%
    - DecisionTreeClassifier + BaggingClassifier: 86%
    - LogisticRegression: 90%
- **Analysis**:



- **Inferences**: (*Reasons according to me why LogisticRegression gave good results and others did not)*
    - Although GradientBoostingClassifier is a powerful algorithm for binary classification the likely reason for why it failed could have been due to imbalance in classes where the number of samples belonging to class 0 and class 1 are heavily imbalanced. This can lead to GradientBoostingClassifier not training properly on the class with fewer samples. Also another issue could be the fact that GradientBoostingClassifier can easily overfit the data if the number of trees is too large or the depth of the tree is too large.
    - In the case of RandomForestClassifier, RandomForestClassifier is a powerful binary classification algorithm. But like GradientBoostingClassifier it had not given higher accuracy due to imbalance in classes where the number of samples belonging to class 0 and class 1 were heavily imbalanced. This can lead to RandomForestClassifier overfitting the class with the largest number of samples. Also another reason could be that RandomForestClassifier cannot distinguish between important features and irrelevant features from the training data and hence could perform poorly in the binary classification task.
    - In the case of GaussianNB (Naive Bayes Classifier) one of the fundamental issues is that it assumes that the features are independent of each other. This may not be the case always and hence it leads to a not so optimal accuracy. Also

another issue like GradientBoostingClassifier and RandomForestClassifier, GaussianNB fails in the case that the number of samples belonging to class 0 and class 1 are heavily imbalanced.

- DecisionTreeClassifier is a widely used binary classification algorithm. But it has a few issues. Mainly, it tends to overfit the data if the depth of the tree is large or if there are too many features in the dataset. There is also a lot of Hyperparameter tuning that needs to be done in order for the DecisionTreeClassifier to perform. This was hard to do due to the limited number of submissions that could be done on the Kaggle platform.
- DecisionTreeClassifier combined with BaggingClassifier is powerful for binary classification. However it fails to provide optimal accuracy because of the similar reasons as RandomForestClassifier. These could include sample imbalances in the number of samples belonging to class 0 and class 1. Also it tends to overfit the data if the depth of the tree is too large.
- LogisticRegression gives me the best accuracy because of the following reasons: It is able to handle outliers or noise in the dataset very well. It works really well for small datasets (the dataset that we are given is small). Also, LogisticRegression includes regularization techniques such as L1 and L2 regularization, which can help prevent overfitting and improve the performance of the model. Hence for this dataset LogisticRegression gives us the best accuracy of 90%.