**Stevens Insitute of Technology**

**Course: CS559WS Machine Learning: Fundamentals and Applications**

**Final Project Report**

**Prepared by :**

**Kshitij Patil 20012527**

**Sanjeet Jain 20012768**

**Savil Robert Dsouza 20008865**

**Professor: In Suk Jang**

**Contents**

* Introduction
* Workflow Summary
* Model Results
* Individual Contributions:
* Final Rank and Kaggle Competition Results
* Conclusion

**Introduction**

**Dataset problem statement:**

In the year 2912, the Spaceship Titanic, carrying 13,000 passengers on its maiden voyage to new habitable exoplanets, collided with a hidden spacetime anomaly near Alpha Centauri. About half of the passengers vanished into an alternate dimension. Data science skills are now needed to solve this cosmic mystery and potentially rescue the missing individuals.

To help rescue crews and retrieve the lost passengers, we made model to predict which passengers were transported by the anomaly using records recovered from the spaceship’s damaged computer system.

**Data used:** [Spaceship Titanic Kaggle](https://www.kaggle.com/competitions/spaceship-titanic/overview)

Personal records for about two-thirds (~8700) of the passengers

* PassengerId - A unique Id for each passenger. Each Id takes the form gggg\_pp where gggg indicates a group the passenger is travelling with and pp is their number within the group. People in a group are often family members, but not always.
* HomePlanet - The planet the passenger departed from, typically their planet of permanent residence.
* CryoSleep - Indicates whether the passenger elected to be put into suspended animation for the duration of the voyage. Passengers in cryosleep are confined to their cabins.
* Cabin - The cabin number where the passenger is staying. Takes the form deck/num/side, where side can be either P for *Port* or S for *Starboard*.
* Destination - The planet the passenger will be debarking to.
* Age - The age of the passenger.
* VIP - Whether the passenger has paid for special VIP service during the voyage.
* RoomService, FoodCourt, ShoppingMall, Spa, VRDeck - Amount the passenger has billed at each of the *Spaceship Titanic*'s many luxury amenities.
* Name - The first and last names of the passenger.
* Transported - Whether the passenger was transported to another dimension. This is the target, the column you are trying to predict.

Our workflow for the class competition consisted of the following steps:

**Data Preprocessing**: We collectively performed data preprocessing on the provided datasets, including train\_.csv, validation\_.csv, and test\_.csv. This involved handling missing values, treating outliers, and addressing any inconsistencies in the data. We utilized relevant libraries/packages to ensure the data was clean and ready for analysis.

**Model Training:** Each team member took responsibility for training multiple models using both parametric and non-parametric algorithms. To avoid duplication, we ensured that each team member selected unique algorithms for their models. We employed the GridSearchCV technique to perform hyperparameter tuning for each model, aiming to identify the optimal combination of hyperparameters that yielded the best performance.

The models we trained include:

* Logistic Regression
* GaussianNB
* Linear Discriminant Analysis
* Decision Tree Classifier
* Random Forest Classifier
* Gradient Boosting Classifier

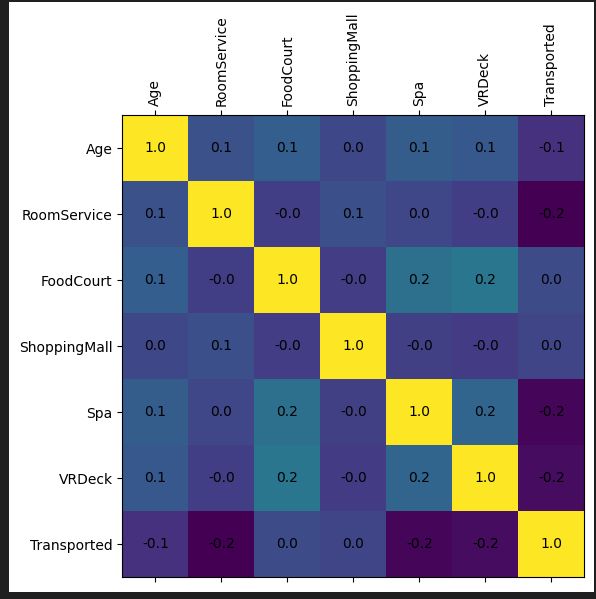
**Stacking Model:** For the final model, we implemented the stacking method in two ways. Firstly we combined the predictions from the previously trained models to create a meta-model. The stacking ensemble technique allowed us to leverage the strengths of multiple base models, enhancing the overall prediction accuracy. We assigned weights (w\_i) to each model and calculated the final prediction (ŷ) using the formula:

secondly we also used the StackingClassifier Model in sklearn as well.

**Model Evaluation:** As a team, we evaluated the performance of each model on the test dataset using appropriate evaluation metrics, with accuracy being the primary metric of interest. We reported the accuracy of each model, including the final stacked model, on the test dataset. This assessment helped us gauge the effectiveness of the models and identify the best-performing ones.

**Documentation and Reporting**: Throughout the workflow, we maintained detailed documentation of the entire process. This included recording the preprocessing steps, model selection, hyperparameter tuning using GridSearchCV, and the implementation of the stacking methodology. We also documented our individual contributions and responsibilities. Finally, we prepared a comprehensive report summarizing the workflow, results, and findings of the project.

By following this collaborative workflow, we ensured thorough data preprocessing, trained multiple models using GridSearchCV, implemented the stacking method for the final model, and evaluated the models based on the accuracy of the test dataset. The documentation and reporting helped us keep track of the process and present a comprehensive overview of the project.

**Correlation Matrix Before Processing**

- The matrix shows the correlation coefficients between different variables in the dataset.

- The variables listed are Age, RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, and Transported.

- The values in the matrix range from -1 to 1, where -1 represents a perfect negative correlation, 0 represents no correlation, and 1 represents a perfect positive correlation.

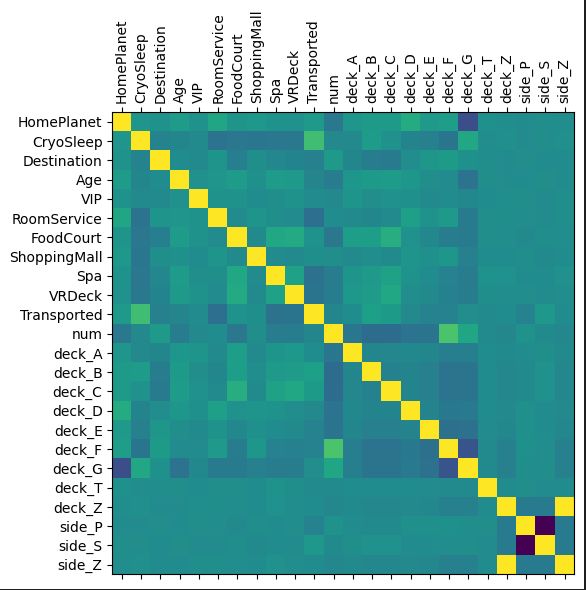
- There are some moderate positive correlations between some of the variables, such as a positive correlation of 0.24 between FoodCourt and VRDeck, and a positive correlation of 0.22 between Spa and FoodCourt.

- There are also some weaker positive correlations, such as a correlation of 0.13 between Age and FoodCourt.

- There are some moderate negative correlations as well, such as a negative correlation of -0.24 between RoomService and Transported, and a negative correlation of -0.22 between Spa and Transported.

- There is also a weaker negative correlation of -0.07 between Age and Transported

**Correlation Matrix After Processing**



* "Transported" is weakly positively correlated with "HomePlanet" and "Age" variables with correlation coefficients of 0.10 and -0.07, respectively. This suggests that individuals from certain home planets and older individuals are slightly more likely to be transported.
* "Transported" is weakly negatively correlated with "CryoSleep" and "num" variables with correlation coefficients of -0.40 and -0.13, respectively. This suggests that individuals who are in cryosleep and those with higher "num" (presumably representing a higher class) are less likely to be transported.
* "Transported" is weakly negatively correlated with "VIP" and "Destination" variables with correlation coefficients of -0.02 and -0.11, respectively. These correlations are quite weak and suggest that these variables are not strong predictors of whether an individual is transported or not.
* "Transported" is strongly negatively correlated with "RoomService" and "ShoppingMall" variables with correlation coefficients of -0.24 and -0.19, respectively. This suggests that individuals who have used the room service and the shopping mall amenities are less likely to be transported.

There is no significant correlation between "Transported" and the remaining variables in the correlation matrix.

[Correlation matrix data File](corrmatrix.csv)

Confusion Matrix

**Model Results**

**Model 1: Logistic Regression**

We have used Logistic regression model in this project for following reasons:

Binary Classification: Logistic regression is well-suited for binary classification tasks where the goal is to predict one of two classes. In this code, it is being used to classify instances into one of the two classes.

Probability Estimation: Logistic regression provides a probability estimate for instance belonging to the positive class. By using the predict function, the model predicts the class labels (0 or 1), and these predictions can be interpreted as probabilities.

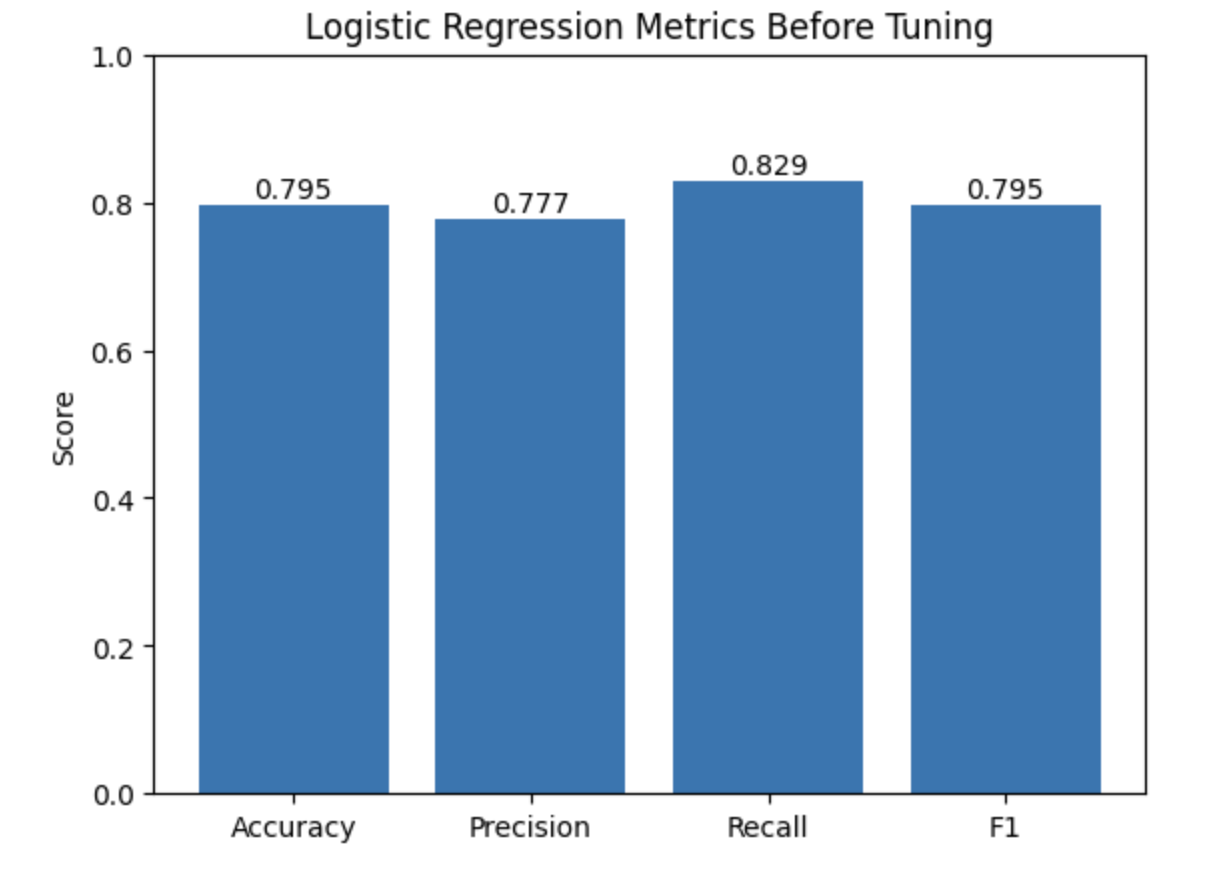
Evaluation Metrics: Logistic regression is commonly evaluated using metrics such as accuracy, precision, recall, and F1-score. These metrics assess the performance of the model in terms of correctly classifying instances and capturing the true positive and negative cases.

Interpretability: Logistic regression coefficients can be interpreted to understand the impact of each feature on the predicted outcome. By analyzing the coefficients, one can gain insights into which features are more influential in determining the class label.

Baseline Model: Logistic regression is often used as a baseline model to compare the performance of other, more complex models. It provides a simple and interpretable benchmark against which other models can be evaluated.

Efficiency: Logistic regression is a computationally efficient algorithm, especially when dealing with a large number of features. It can handle datasets with a high dimensionality without incurring excessive computational costs.

Validation -- Accuracy: 0.783 / Precision: 0.737 / Recall: 0.89 / f1: 0.78  
Test -- Accuracy: 0.794 / Precision: 0.746 / Recall: 0.891 / f1: 0.792



A picture containing text, screenshot, diagram, number

Description automatically generated

A higher recall after tuning the model with GridSearch suggest that the model is performing well in terms of correctly identifying positive instances (high recall)

**Model 2: Gaussian Naive Bayes (GaussianNB)**

We have used Gaussian NB model for following reasons:

Assumption of Gaussian Distribution: GaussianNB assumes that the features follow a Gaussian distribution. If the features in the dataset have a continuous or numerical nature and approximately follow a Gaussian distribution, GaussianNB can be an appropriate choice.

Independence Assumption: GaussianNB assumes that the features are independent of each other given the class variable. This assumption simplifies the modeling process and can be effective in situations where the features are indeed independent or have a weak correlation.

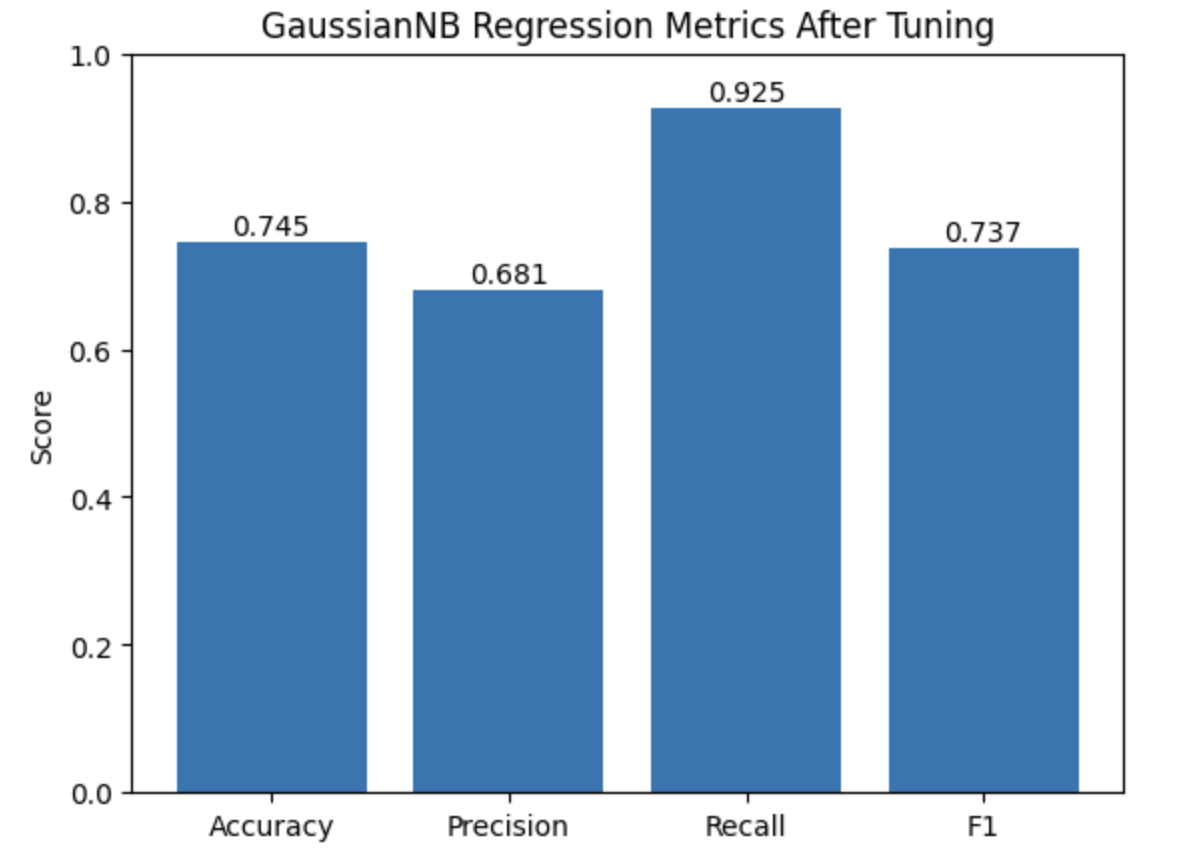
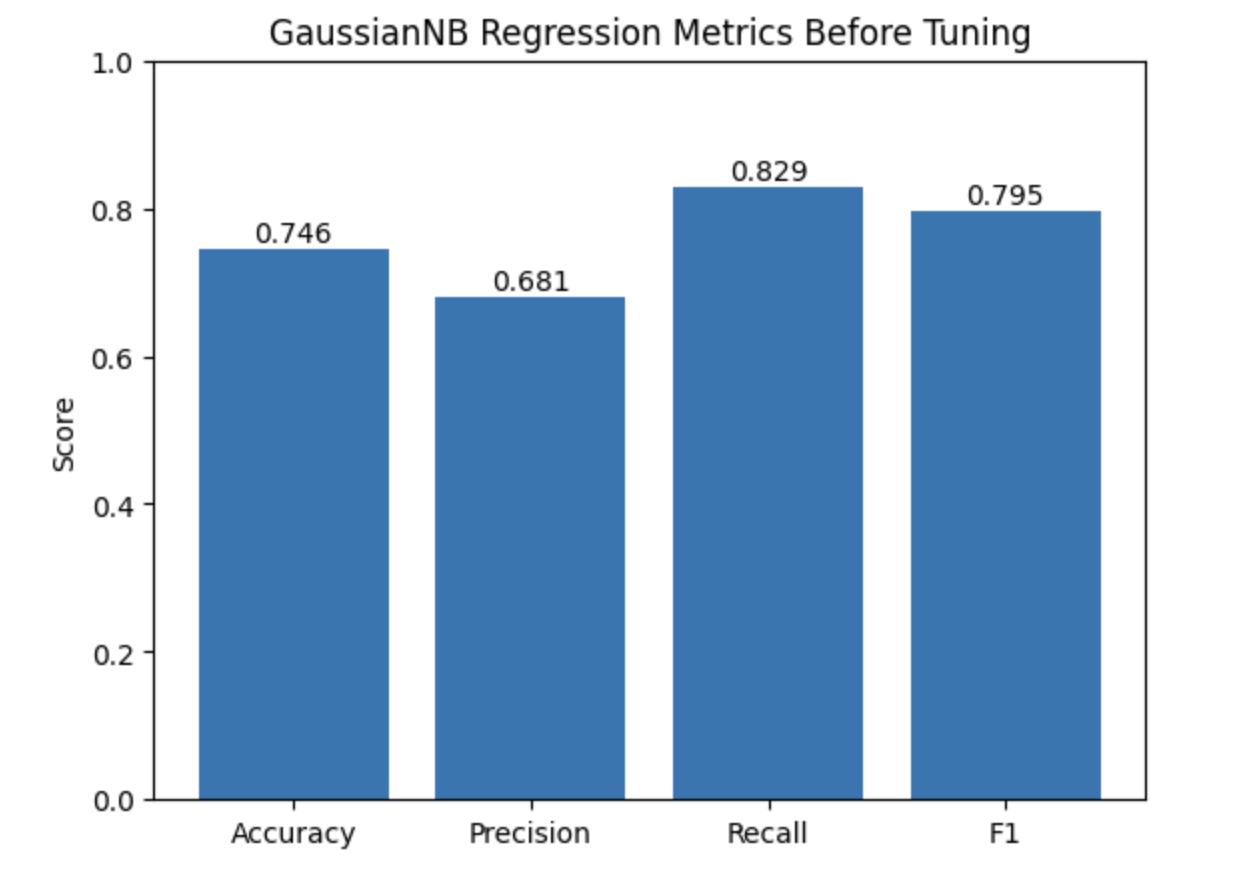
Probabilistic Classification: GaussianNB provides probabilistic classification by estimating the likelihood of an instance belonging to each class based on the observed feature values. The predict function of GaussianNB returns the predicted class labels, and these predictions can be interpreted as probabilities.

Speed and Efficiency: GaussianNB is a fast and computationally efficient algorithm. It has a relatively low training time and requires fewer computational resources compared to more complex models.

Handling Continuous Features: GaussianNB handles continuous features well due to its assumption of a Gaussian distribution. It calculates the mean and standard deviation of each feature for each class during training and uses this information to estimate probabilities.

Interpretability: GaussianNB provides interpretability in terms of feature importance. By analyzing the mean and standard deviation of each feature for each class, one can gain insights into which features are more influential in determining the class label.

Validation -- Accuracy: 0.726 / Precision: 0.675 / Recall: 0.884 / f1: 0.717  
Test -- Accuracy: 0.745 / Precision: 0.681 / Recall: 0.925 / f1: 0.737



A higher recall after tuning the model with GridSearch suggest that the model is performing well in terms of correctly identifying positive instances (high recall)

**Model 3: Linear Discriminant Analysis (LDA)**

We have used LDA model for following reasons:

Dimensionality Reduction: LDA can be used as a dimensionality reduction technique. It seeks to find a linear combination of features that maximizes the separation between different classes while minimizing the variance within each class. By projecting the data onto this lower-dimensional space, LDA can potentially improve the performance of classification models by reducing the impact of irrelevant or redundant features.

Classification: LDA is also a classification algorithm. It models the distribution of features in each class and uses Bayes' theorem to calculate the posterior probabilities of class membership. By comparing these probabilities, LDA assigns a class label to each instance. It assumes that the features are normally distributed and that the class covariances are equal.

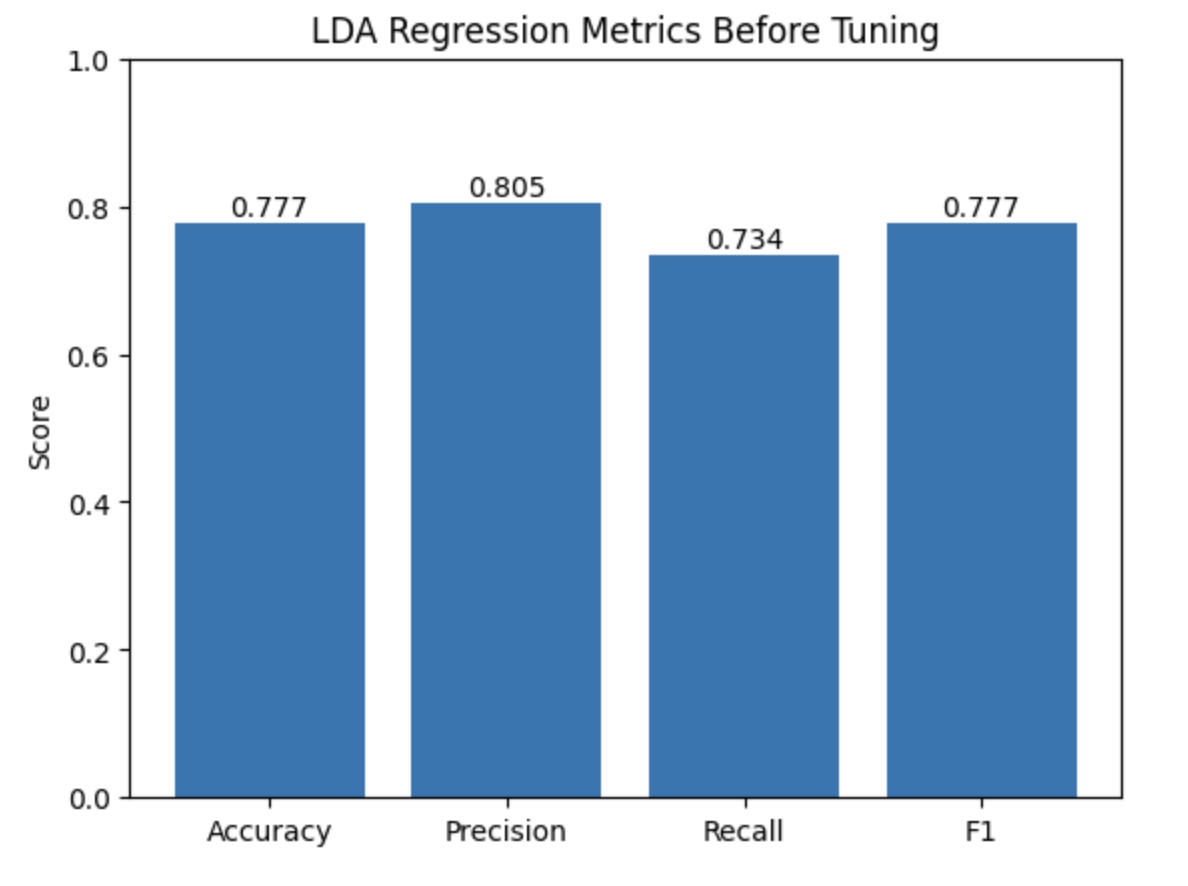
Handling Multicollinearity: LDA performs well in situations where there is multicollinearity among the features. It can handle situations where the features are highly correlated by considering the relationship between the features and the target variable.

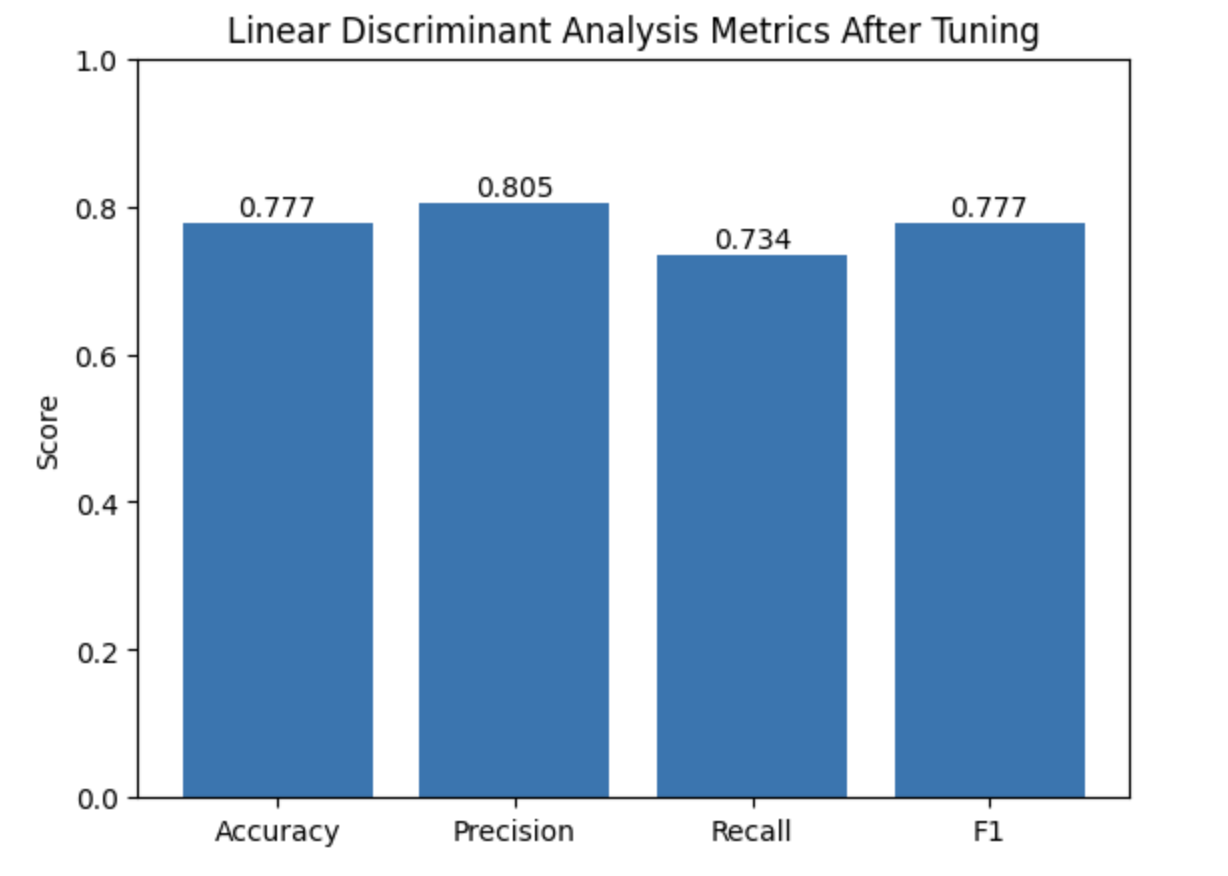
Interpretability: LDA provides interpretability by allowing us to analyze the coefficients or discriminant vectors that are derived during the training process. These coefficients indicate the contribution of each feature to the separation between classes, enabling us to identify the most discriminative features.

Balanced Classes: LDA assumes that the classes have approximately equal covariance matrices. It works best when the classes are balanced or have similar within-class variances. If the classes are imbalanced, LDA may not perform as well.

Assumption of Linearity: LDA assumes that the decision boundaries between classes are linear. If the relationship between the features and the target variable is complex and nonlinear, other algorithms such as kernel-based methods or nonlinear classifiers may be more suitable.

Validation -- Accuracy: 0.76 / Precision: 0.787 / Recall: 0.722 / f1: 0.76  
test -- Accuracy: 0.777 / Precision: 0.805 / Recall: 0.734 / f1: 0.777





**Model 4: Decision Tree Classifier (DTC)**

We have used DTC for following reasons:

Interpretable Decision Rules: Decision trees are easy to interpret and understand. The model builds a tree-like structure where each internal node represents a feature, each branch represents a decision rule, and each leaf node represents a class label. This makes it easier to explain the logic behind the classification decisions made by the model.

Nonlinear Relationships: Decision trees can capture nonlinear relationships between features and the target variable. They can handle complex decision boundaries and interactions among features without requiring explicit feature engineering or transformation.

Feature Importance: Decision trees provide a measure of feature importance, which indicates the relative importance of each feature in the classification process. This information can be useful for feature selection, identifying the most relevant features, and gaining insights into the underlying data.

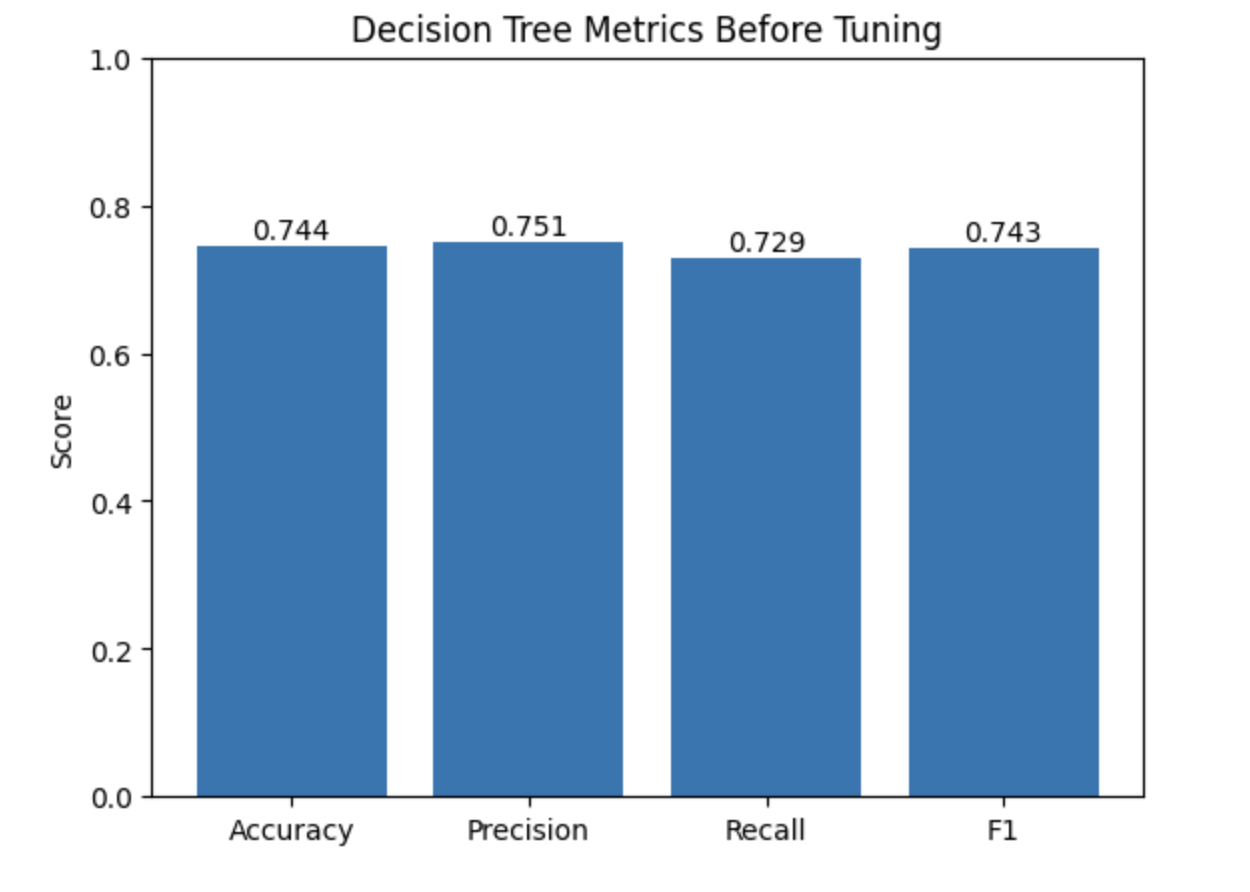
Handling Missing Values and Outliers: Decision trees can handle missing values and outliers by using surrogate splits or by assigning missing values to the most common class in the training set. This can be advantageous when dealing with datasets that have missing or incomplete information.

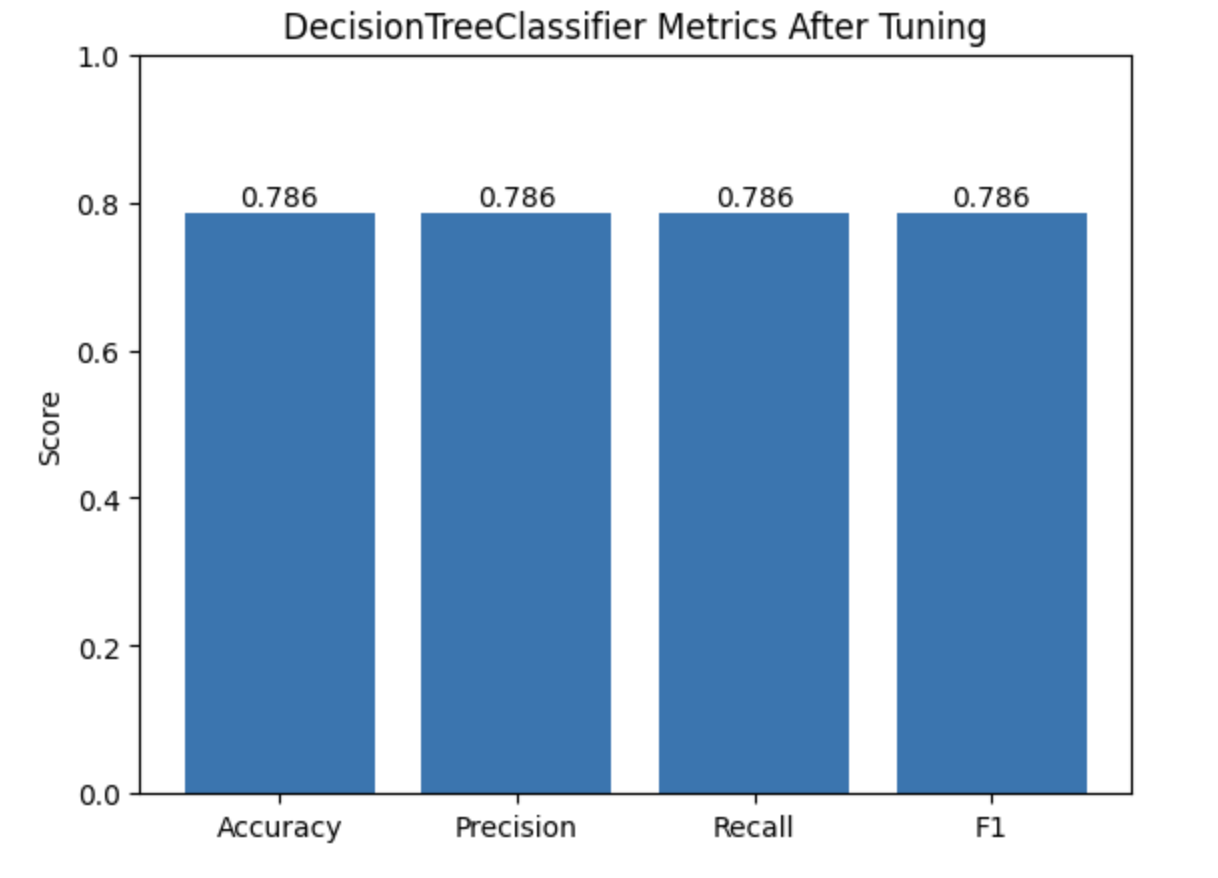
No Assumptions about Data Distribution: Decision trees do not assume any specific data distribution. They can work well with both categorical and numerical features without making assumptions about the underlying data.

Ensemble Methods: Decision trees can be used as building blocks for ensemble methods such as random forests and gradient boosting. These ensemble methods combine multiple decision trees to improve the overall predictive performance and reduce overfitting.

Training Efficiency: Decision trees can be trained efficiently, especially when the dataset is not too large. The training time complexity of decision trees is generally logarithmic in the number of training instances.

Validation -- Accuracy: 0.766 / Precision: 0.76 / Recall: 0.788 / f1: 0.766  
test -- Accuracy: 0.786 / Precision: 0.786 / Recall: 0.786 / f1: 0.786





Overall perfomance was slightly increased after tuning the model

**Model 5: Random Forest Classifier (RFC)**

We have used rfc for following reason:

Ensemble Learning: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. By aggregating the predictions of multiple trees, it reduces the variance and improves the overall performance compared to a single decision tree.

Improved Generalization: Random Forests are less prone to overfitting compared to individual decision trees. By combining predictions from different trees and considering random subsets of features for each tree, Random Forests reduce the risk of overfitting and improve generalization on unseen data.

Feature Importance: Random Forests provide a measure of feature importance based on how much each feature contributes to the overall prediction accuracy. This information can be used for feature selection, identifying the most influential features, and gaining insights into the dataset.

Handling Nonlinear Relationships: Random Forests can effectively capture nonlinear relationships between features and the target variable. They can handle complex decision boundaries and interactions among features, making them suitable for datasets with nonlinear relationships.

Robustness to Noisy Data: Random Forests are robust to noisy data and outliers. The majority voting mechanism of the ensemble helps to reduce the impact of individual noisy or erroneous predictions made by individual trees.

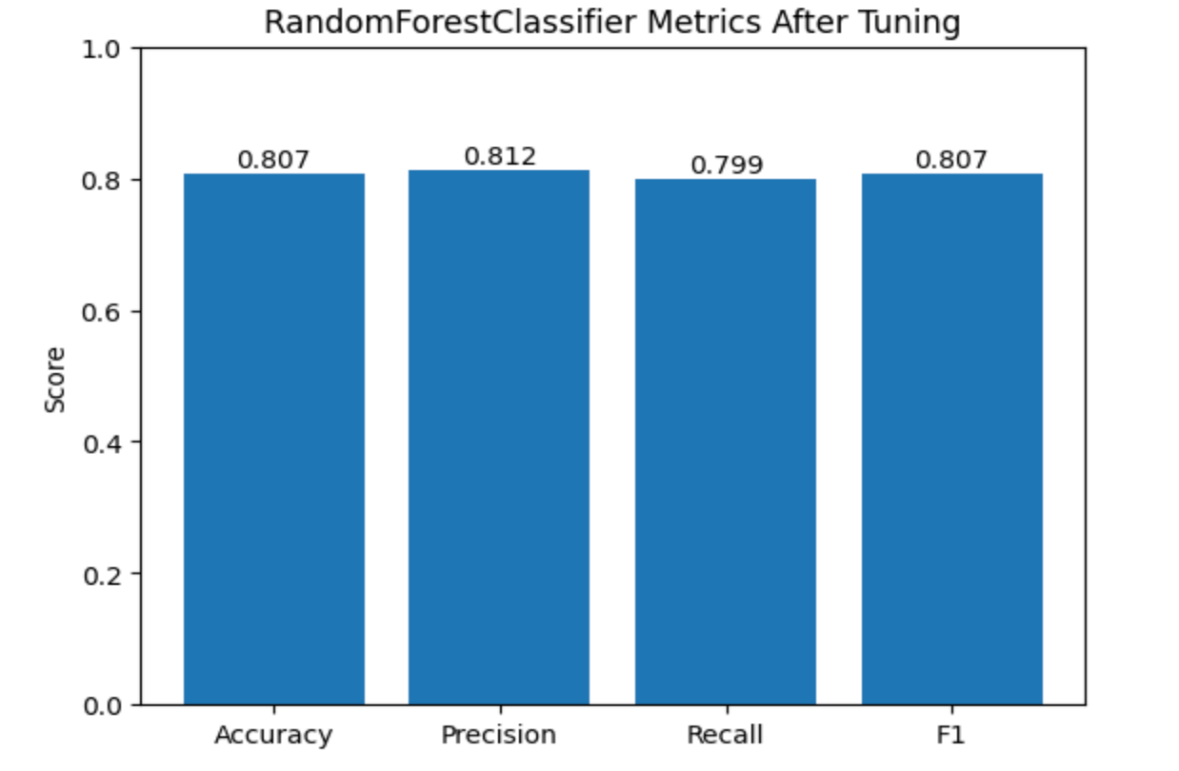
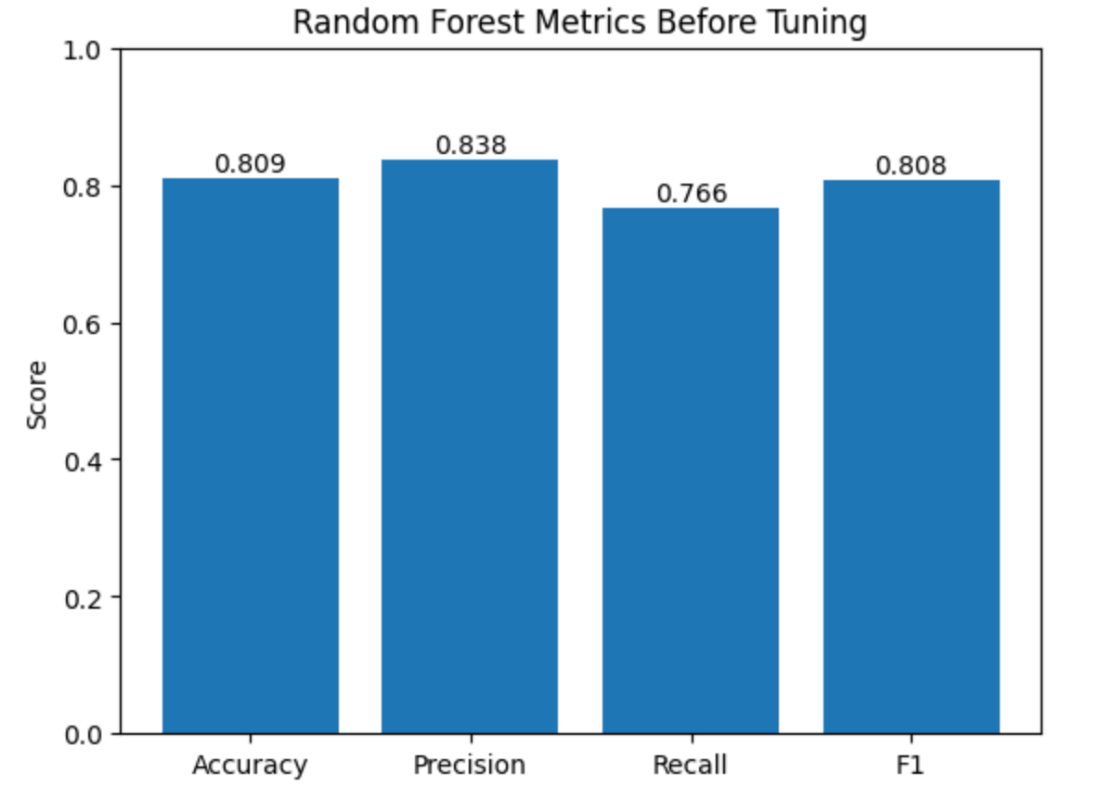
Handling Missing Values: Random Forests can handle missing values in the dataset by imputing missing values based on the available information in other features. This reduces the need for extensive data preprocessing or imputation techniques.

Scalability: Random Forests can handle large datasets with a large number of features efficiently. The training process can be parallelized, and predictions can be made quickly, making them scalable for real-world applications.

No Assumptions about Data Distribution: Random Forests do not make assumptions about the distribution of the data or the relationship between features. They can handle both categorical and numerical features without requiring data transformation or normalization.

Validation -- Accuracy: 0.776 / Precision: 0.791 / Recall: 0.759 / f1: 0.776

test -- Accuracy: 0.807 / Precision: 0.812 / Recall: 0.799 / f1: 0.807



A higher recall after tuning the model with GridSearch suggest that the model is performing well in terms of correctly identifying positive instances (high recall)

**Model 6: Gradient Boosting Classifier**

We have used gradient boosting classifier for following reason:

Boosting Technique: Gradient Boosting is a boosting technique that combines multiple weak learners (decision trees) to create a strong predictive model. It builds the model in an iterative manner, where each subsequent model tries to correct the mistakes of the previous models, leading to improved performance.

High Predictive Power: Gradient Boosting is known for its high predictive power and ability to capture complex relationships in the data. By combining multiple weak learners, it can model nonlinear relationships, interactions, and complex decision boundaries effectively.

Addressing Bias-Variance Trade-off: Gradient Boosting strikes a balance between bias and variance, which helps in reducing both underfitting and overfitting. It typically performs better than individual decision trees by reducing the bias and variance errors through ensemble learning.

Feature Importance: Gradient Boosting provides a measure of feature importance based on how much each feature contributes to the overall improvement of the model's performance. This information can be used for feature selection, identifying important features, and gaining insights into the dataset.

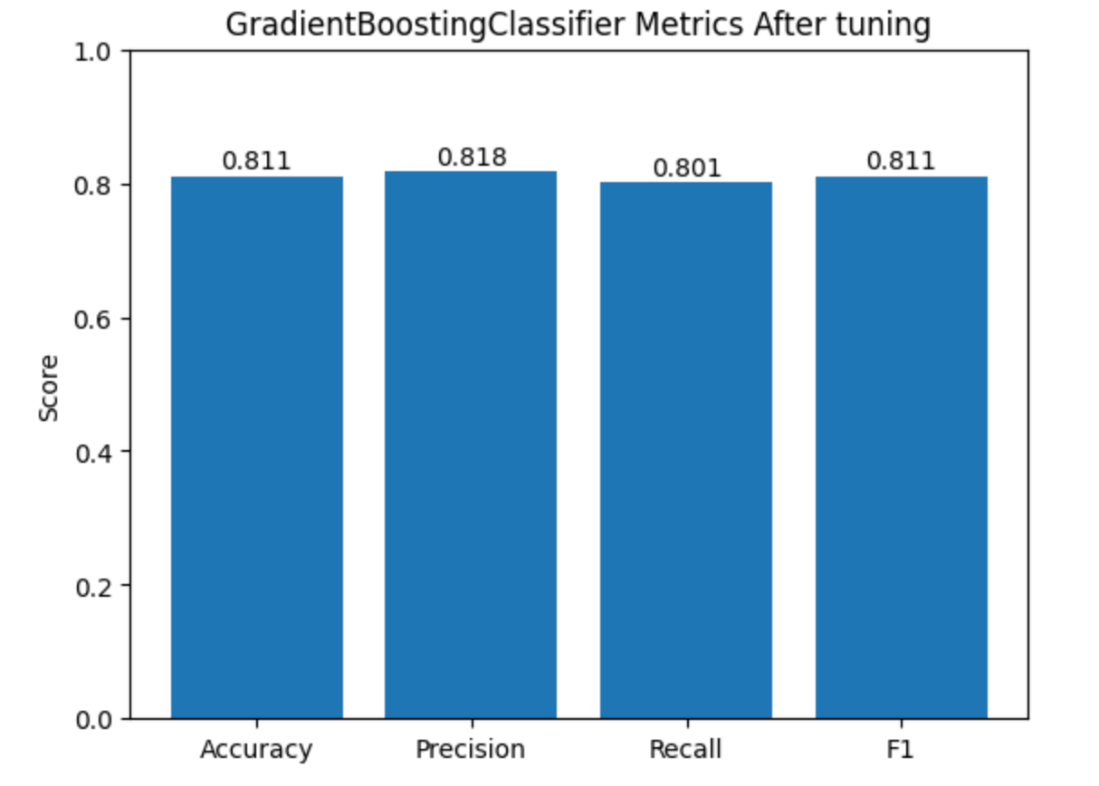
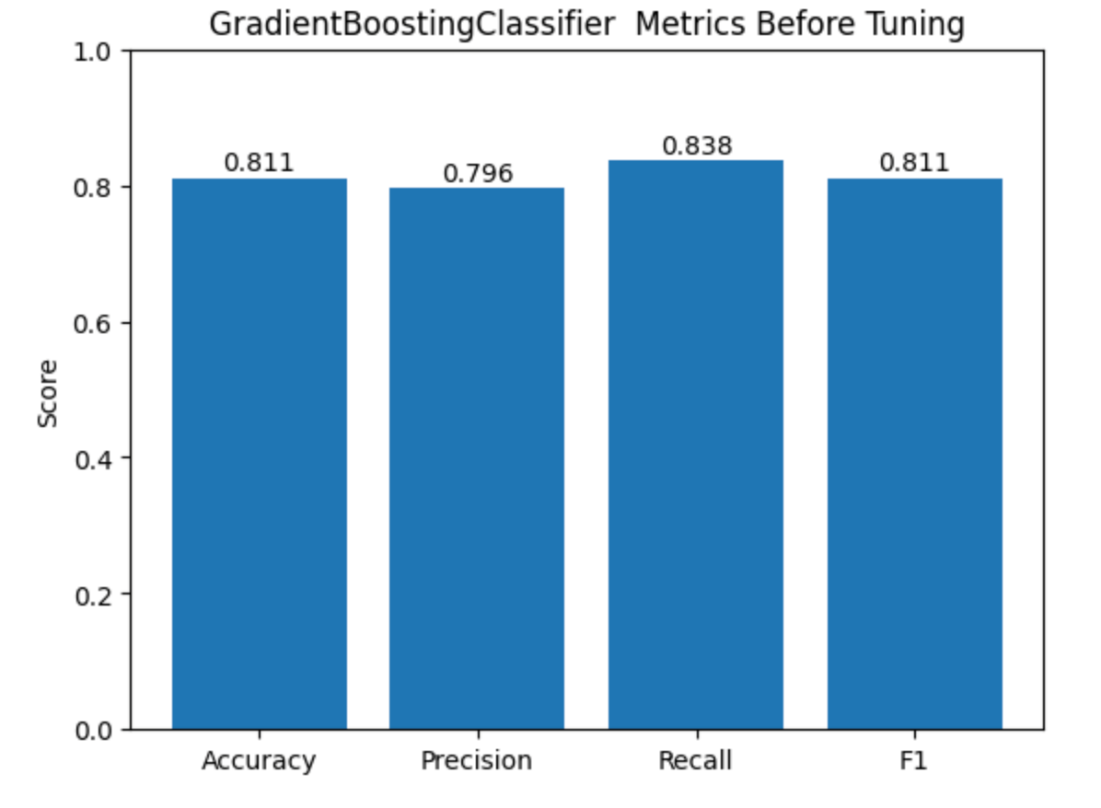
Handling Missing Values: Gradient Boosting can handle missing values in the dataset by using surrogate splits. It can effectively utilize the available information from other features to make predictions even in the presence of missing values.

Robustness to Noisy Data: Gradient Boosting is robust to noisy data and outliers. It can adapt and adjust the weights of the observations during the training process, reducing the impact of noisy data points on the final model.

Ensemble Learning: Gradient Boosting combines multiple weak learners to make predictions, resulting in improved accuracy and robustness. The ensemble nature of Gradient Boosting helps in capturing diverse patterns and reducing the impact of individual weak learners.

Scalability: Gradient Boosting is scalable and can handle large datasets with a large number of features. It can be parallelized to speed up the training process and make predictions efficiently.

Validation -- Accuracy: 0.789 / Precision: 0.805 / Recall: 0.771 / f1: 0.789 test -- Accuracy: 0.811 / Precision: 0.818 / Recall: 0.801 / f1: 0.811



The tuned model has a better balance between accuracy precision and recall

We have also done stacking ensemble for following reason:

Stacking Ensemble: Stacking is an ensemble learning technique that combines multiple base models (or estimators) with a meta-model to make predictions. It leverages the strengths of different models and aims to improve overall prediction performance.

Stacking Classifier: The Stacking Classifier is a specific implementation of stacking in scikit-learn. It takes a list of base estimators (Logistic Regression, GaussianNB, Linear Discriminant Analysis, Decision Tree Classifier, Random Forest Classifier, and Gradient Boosting Classifier) and a final estimator (Logistic Regression) as inputs. The base estimators make individual predictions, and the final estimator (also known as the meta-model) combines those predictions to make the final prediction.

Meta-Model Training: The stacking classifier trains the base estimators on the training set (X\_train, y\_train). Each base estimator learns to make predictions based on the input features. Then, the stacking classifier combines the predictions from these base estimators to create a new dataset (referred to as the "meta-features") for the final estimator.

Cross-Validation: The stacking classifier performs cross-validation (cv=5) during training. It splits the training set into multiple folds and trains each base estimator on a subset of the training data. This helps to assess the performance and generalization of the stacking ensemble.

Stack Method: The parameter "stack\_method='auto'" indicates that the stacking classifier automatically determines the method to use for combining the base estimators' predictions. It can be set to 'auto', 'predict\_proba', or 'decision\_function' depending on the type of predictions the base estimators provide.

Parallelization: The stacking classifier utilizes parallel processing (n\_jobs=-1) to train the base estimators and the final estimator in parallel, taking advantage of multiple CPU cores to speed up the training process.

Prediction and Evaluation: After training the stacking classifier, it makes predictions on the validation set (X\_val) and the test set (X\_test). The accuracy of the meta-model's predictions is evaluated using the accuracy\_score function and printed as output.

Using Stacking Classifier Model :

Accuracy of meta model on validation set: 0.795  
Accuracy of meta model on test set: 0.815

Using Stacking formula

Accuracy of alternative stacking model on test data 0.814

A picture containing text, screenshot, display, software

Description automatically generated

The resulting plot allows us to visualize the relative importance of each feature in the model. A higher point on the y-axis indicates that the corresponding feature is more important in the prediction process. By comparing the feature importances across different estimators, we can also get an idea of how consistent the feature importances are across different models.

**Conclusion:**

|  |  |
| --- | --- |
| Model | Accuracy on Test Set |
| Logistic Regression | 0.794 |
| Gaussian Naïve Bayes | 0.745 |
| LDA | 0.777 |
| Decision Tree Classifier | 0.786 |
| Random Forest Classifier | 0.807 |
| Gradient Boosting Classifier | 0.811 |
| Stacked Model #1  Stacked Model #2 | 0.815  0.814 |

Insert our screenshot here

**Final Result and Kaggle Competition results:**

Our rank in Kaggle competition was 854 on may 5th with a score of 79.73% using XGBoost.

After tuning the model more in our later days and using a stacking model we were able to achieve a score if 0.79845 on the Kaggle competition by running the code on the Kaggle jupyter notebook.

The screenshot is provided below:

