Imports

1. **pandas (pd):** Used for data manipulation and analysis, providing data structures like dataframes.
2. **numpy (np):** Essential for numerical operations and handling arrays, matrices, and mathematical functions.
3. **GridSearchCV:** From scikit-learn, used for hyperparameter tuning through exhaustive search over specified parameter values for an estimator.
4. **MinMaxScaler:** Another scikit-learn module, used for scaling numerical features to a specified range (usually between 0 and 1).
5. **matplotlib.pyplot (plt):** A plotting library, commonly used for visualizing data and model performance.
6. **seaborn (sns):** Built on top of Matplotlib, seaborn is used for statistical data visualization, providing a high-level interface for drawing attractive and informative statistical graphics.
7. **confusion_matrix, mean_squared_error, r2_score, mean_absolute_error:** Evaluation metrics from scikit-learn for assessing classification and regression model performance.
8. **train_test_split:** Function from scikit-learn for splitting datasets into training and testing sets.
9. **RandomForestRegressor, RandomForestClassifier, AdaBoostClassifier, AdaBoostRegressor, GradientBoostingClassifier, GradientBoostingRegressor:** Various machine learning models from scikit-learn, including Random Forest, AdaBoost, and Gradient Boost, for both classification and regression tasks.
10. **accuracy_score:** Metric for measuring the accuracy of classification models, particularly useful for evaluating the performance of classifiers.

Predefined Functions

Firstly, we defined two functions to evaluate the performance of each model. One function calculates classification performance metrics (Accuracy, Precision, Recall, F1-Score) and the other one calculates regression performance metrics (Mean Squared Error, Mean Absolute Error, Root Squared Error). We simply defined a function, for less code repetitiveness.

First Dataset – bankloan.csv

Preprocessing Steps

We, firstly, check for any null values in the entire dataset, which shows that there are none. Secondly, we normalize/ scale the ZIP.CODE column to values between 0 and 1, using the MinMaxScaler, because the column contains very large values. Finally, we will drop any unnecessary columns, which leads to dropping the "ID" column, since it should not be included in the mashine learning algorithms.

Random Forest Model Implementation

Hyperparameter Tuning

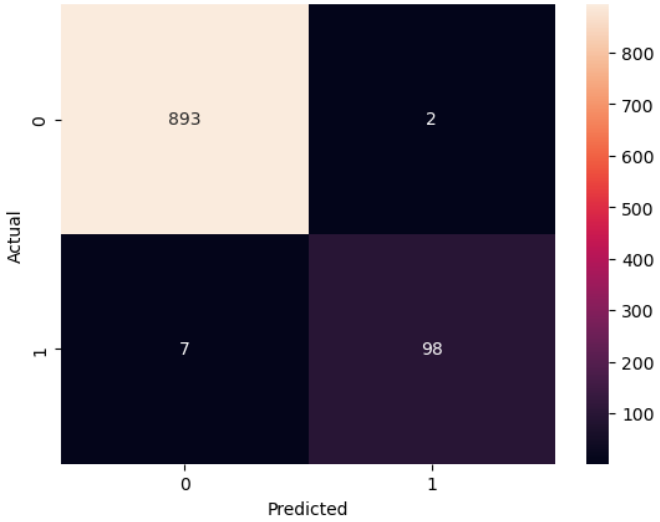
In our team, we conducted hyperparameter tuning for the RandomForestClassifier on the first dataset using grid search. We defined a parameter grid with variations for key hyperparameters and initialized a base model with default settings, incorporating a random_state for reproducibility.

The grid search model, instantiated through GridSearchCV, systematically explored hyperparameter combinations. Upon fitting the model to the training data, we printed the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the RandomForestClassifier on the first dataset includes setting bootstrap to False, max_depth at 20, max_features using 'sqrt', min_samples_leaf at 1, min_samples_split at 2, and n_estimators at 300. This resulted in a highly effective classifier, achieving an impressive score of approximately 0.9855 on the training data.

Results – Evaluation of the Model (on the test data)

```
Best Classifier Accuracy: 0.991
Best Classifier Precision: 0.98
Best Classifier Recall: 0.9333333333333333
Best Classifier F1 Score: 0.9560975609756097
```



AdaBoost Model Implementation

Hyperparameter Tuning

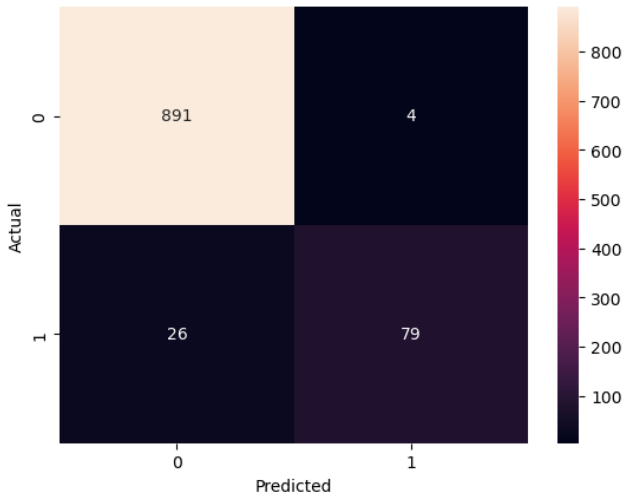
In our AdaBoost model implementation, we conducted hyperparameter tuning using grid search on the first dataset. The parameter grid (param_gridABC) encompassed variations for 'n_estimators' and 'learning_rate,' key hyperparameters crucial for optimizing AdaBoostClassifier performance. Initializing a base model (abcModel) with a random_state parameter for reproducibility, we instantiated the grid search model (grid_searchABC) using GridSearchCV.

The grid search model systematically explored hyperparameter combinations. After fitting the model to the training data (X_train1, y_train1), we printed the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the AdaBoostClassifier on the first dataset was determined as a learning rate of 0.1 and 200 estimators. This resulted in a highly effective classifier, achieving an accuracy score of 0.97 on the training data.

Results – Evaluation of the Model (on the test data)

```
Best Classifier Accuracy: 0.97
Best Classifier Precision: 0.9518072289156626
Best Classifier Recall: 0.7523809523809524
Best Classifier F1 Score: 0.8404255319148937
```



Gradient Boost Model Implementation

Hyperparameter Tuning

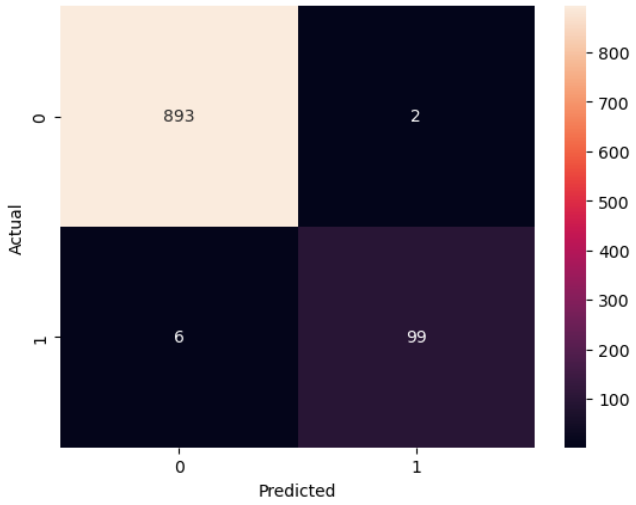
Our Gradient Boost model implementation involved complex hyperparameter tuning using grid search on the first dataset. The parameter grid (param_gridGBC) is constructed to explore variations for key hyperparameters such as 'n_estimators', 'learning_rate', 'max_depth', 'min_samples_split', 'min_samples_leaf', 'subsample', and 'max_features'. Initializing a base model (gbcModel) with a random_state parameter for reproducibility, we instantiate the grid search model (grid_searchGBC) using GridSearchCV.

The grid search model systematically explored hyperparameter combinations. After fitting the model to the training data (X_train1, y_train1), we print the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the GradientBoostingClassifier on the first dataset included a learning rate of 0.3, max depth of 3, 'sqrt' for max features, minimum samples per leaf at 1, minimum samples per split at 10, 100 estimators, and a subsample of 0.8. This results in a highly effective classifier, achieving an accuracy score of 0.992 on the training data.

Results – Evaluation of the Model (on the test data)

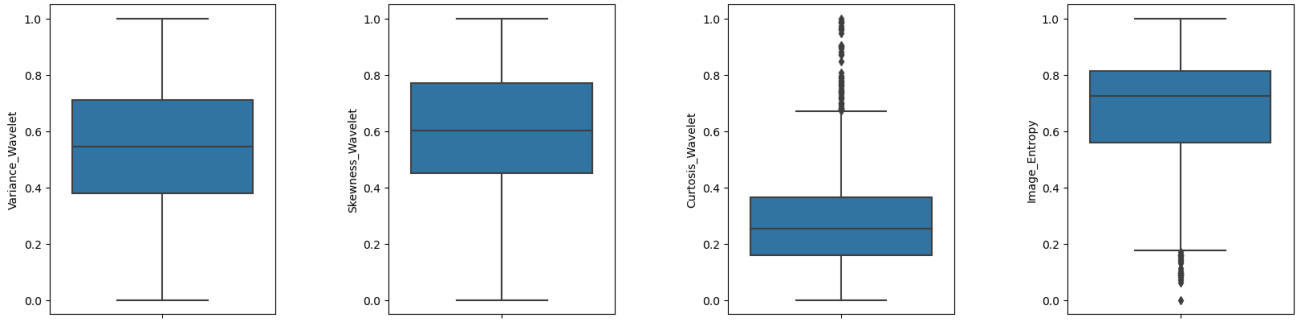
```
Best Classifier Accuracy: 0.992
Best Classifier Precision: 0.9801980198019802
Best Classifier Recall: 0.9428571428571428
Best Classifier F1 Score: 0.9611650485436893
```



Second Dataset - data_banknote_authentication.csv

Preprocessing Steps

We, firstly, check for any null values in the entire dataset, which shows that there are none. Then, we check for non-numeric data in the entire dataset. The dataset is entirely made out of numeric columns, so they should all contain numeric column values. We, then see that there are no non-numeric values. Thirdly, we normalize/ scale all columns except for the label column to values between 0 and 1, using the MinMaxScaler, because some columns contain negative numbers, and we also want all columns here to have the same scale. Lastly, we check for outliers; We find some, which we then remove from the dataset.



Outliers for Columns: Variance_Wavelet, Skewness_Wavelet, Curtosis_Wavelet and Image_Entropy

Random Forest Model Implementation

Hyperparameter Tuning

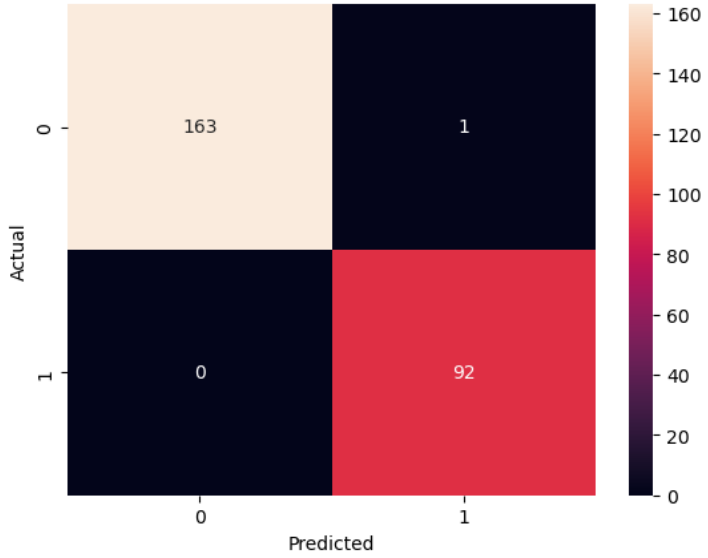
For the Random Forest model on the second dataset, we conducted hyperparameter tuning using grid search. The parameter grid (param_gridRFC2) is designed to explore variations for 'n_estimators', 'max_features', 'max_depth', 'min_samples_split', 'min_samples_leaf', and 'bootstrap'. A base model (rfc2) is initialized with a random_state parameter for reproducibility, and the grid search model (grid_searchRFC2) is instantiated using GridSearchCV.

The grid search model systematically explores hyperparameter combinations. After fitting the model to the training data (X_train2, y_train2), we print the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the RandomForestClassifier on the second dataset involves setting bootstrap to False, max depth at 10, 'sqrt' for max features, minimum samples per leaf at 1, minimum samples per split at 2, and 200 estimators. This configuration results in a highly effective classifier, achieving an accuracy score of 0.992 on the training data.

Results – Evaluation of the Model (on the test data)

```
Best Classifier Accuracy: 0.99609375
Best Classifier Precision: 0.989247311827957
Best Classifier Recall: 1.0
Best Classifier F1 Score: 0.9945945945945946
```



AdaBoost Model Implementation

Hyperparameter Tuning

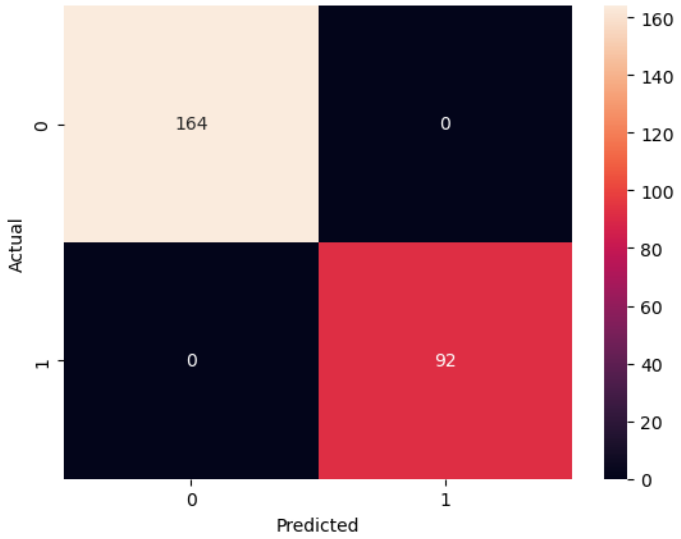
In optimizing the AdaBoostClassifier on the second dataset, we perform hyperparameter tuning using grid search. The parameter grid (param_gridABC2) is crafted to explore variations for 'n_estimators' and 'learning_rate,' hyperparameter. A base model (abcModel2) is initialized with a random_state parameter for reproducibility, and the grid search model (grid_searchABC2) is instantiated using GridSearchCV.

The grid search model systematically explorses hyperparameter combinations. After fitting the model to the training data (X_train2, y_train2), we print the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the AdaBoostClassifier on the second dataset involves a learning rate of 0.1 and 200 estimators, resulting in a highly effective classifier with an accuracy score of 1.0 on the training data.

Results – Evaluation of the Model (on the test data)

```
Best Classifier Accuracy: 1.0
Best Classifier Precision: 1.0
Best Classifier Recall: 1.0
Best Classifier F1 Score: 1.0
```



Gradient Boost Model Implementation

Hyperparameter tuning

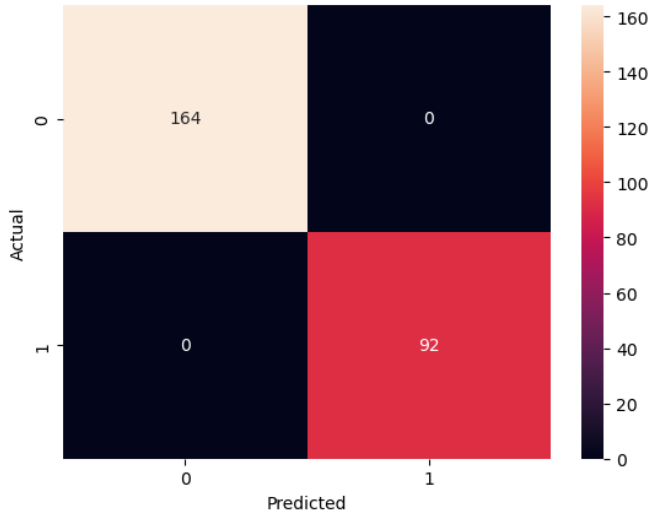
In optimizing the GradientBoostingClassifier on the second dataset, we employ grid search for hyperparameter tuning. The parameter grid (param_gridGBC2) encompasses variations for 'n_estimators', 'learning_rate', 'max_depth', 'min_samples_split', 'min_samples_leaf', 'subsample', and 'max_features.' A base model (gbcModel2) is initialized with a random_state parameter for reproducibility, and the grid search model (grid_searchGBC2) was instantiated using GridSearchCV.

The grid search model systematically explores hyperparameter combinations. After fitting the model to the training data (X_train2, y_train2), we print the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the GradientBoostingClassifier on the second dataset involves a learning rate of 1, max depth of 3, 'sqrt' for max features, minimum samples per leaf at 2, minimum samples per split at 5, 100 estimators, and a subsample of 0.8. This results in a highly effective classifier, achieving an accuracy score of 1.0 on the training data.

Results – Evaluation of the Model (on the test data)

```
Best Classifier Accuracy: 1.0
Best Classifier Precision: 1.0
Best Classifier Recall: 1.0
Best Classifier F1 Score: 1.0
```



Third Dataset – predict taxi fare.csv

Preprocessing Steps

Firstly, we will drop all irrelevant columns, which only includes the “medallion” column as it represents some sort of ID for a data row, and this should not be included in a mashine learning algorithm. Secondly, we take a random sample of size = 8000 data rows from the original dataset because it’s too large.

Now, we do some feature extraction and creation. Out of the “pickup_datetime” column, which contains the exact time of the taxi pickup including seconds to year, we extract the daytime which is simply the hour of the day. And we also extract the weekday, which is represented using numbers from 0 to 6, which mean Monday to Sunday, respectively. Afterwards, we simply drop the “pickup_daytime” column since we created two new columns representing important time data.

Lastly, we check for null values and non-numeric values, which both are not present.

Note: Worth mentioning is that this time around, the dataset label is not a discrete value [0, 1] like in the previous two datasets, but this time it’s a continuous value representing taxi fare money. That is why, this time around, instead of using classification models with classification performance metrics, we used regression models with regression performance metrics.

Random Forest Model Implementation

Hyperparameter Tuning

For the Random Forest Regression model on the third dataset, we conduct hyperparameter tuning using grid search. The parameter grid (param_gridRFR3) is designed to explore variations for 'n_estimators', 'max_features', 'max_depth', 'min_samples_split', 'min_samples_leaf', and 'bootstrap.' A base model (rfr3) is initialized with a random_state parameter for reproducibility, and the grid search model (grid_searchRFR3) is instantiated using GridSearchCV.

The grid search model systematically explores hyperparameter combinations. After fitting the model to the training data (X_train3, y_train3), we print the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the RandomForestRegressor on the third dataset involves setting bootstrap to True, max depth at 20, 'sqrt' for max features, minimum samples per leaf at 2, minimum samples per split at 2, and 128 estimators. This results in a highly effective regressor, achieving a score of 0.818 on the training data.

Results – Evaluation of the Model (on the test data)

```
Out-of-Bag Score: 0.8206060480979982
Mean Squared Error: 12.8456223600785
R-squared: 0.8522800895181022
Mean Absolute Error: 1.5775214773592048
```

AdaBoost Model Implementation

Hyperparameter Tuning

For the AdaBoost Regression model on the third dataset, we perform hyperparameter tuning using grid search. The parameter grid (param_gridABR3) is crafted to explore variations for 'n_estimators' and 'learning_rate,' hyperparameters for optimizing AdaBoostRegressor performance. A base model (abrModel3) is initialized with a random_state parameter for reproducibility, and the grid search model (grid_searchABR3) was instantiated using GridSearchCV.

The grid search model systematically explores hyperparameter combinations. After fitting the model to the training data (X_train3, y_train3), we print the best hyperparameters, best estimator, and the corresponding score.

The optimal configuration for the AdaBoostRegressor on the third dataset involves a learning rate of 0.01 and 200 estimators, resulting in a regressor with a score of 0.7237 on the training data.

Results – Evaluation of the Model (on the test data)

```
Mean Squared Error: 20.284462868168887
R-squared: 0.7667361724433429
Mean Absolute Error: 2.8013100044966803
```

Gradient Boost Model Implementation

Hyperparameter tuning

For the Gradient Boost Regression model on the third dataset, we conduct hyperparameter tuning using grid search. The parameter grid (param_gridGBR3) is designed to explore variations for 'n_estimators', 'learning_rate', 'max_depth', 'min_samples_split', and 'min_samples_leaf.' A base model (gbrModel3) is initialized with a random_state parameter for reproducibility and 'sqrt' for max_features. The grid search model (grid_searchGBR3) is instantiated using GridSearchCV.

The grid search model systematically explores hyperparameter combinations. After fitting the model to the training data (X_train3, y_train3), we print the best hyperparameters, best estimator, and the corresponding score.

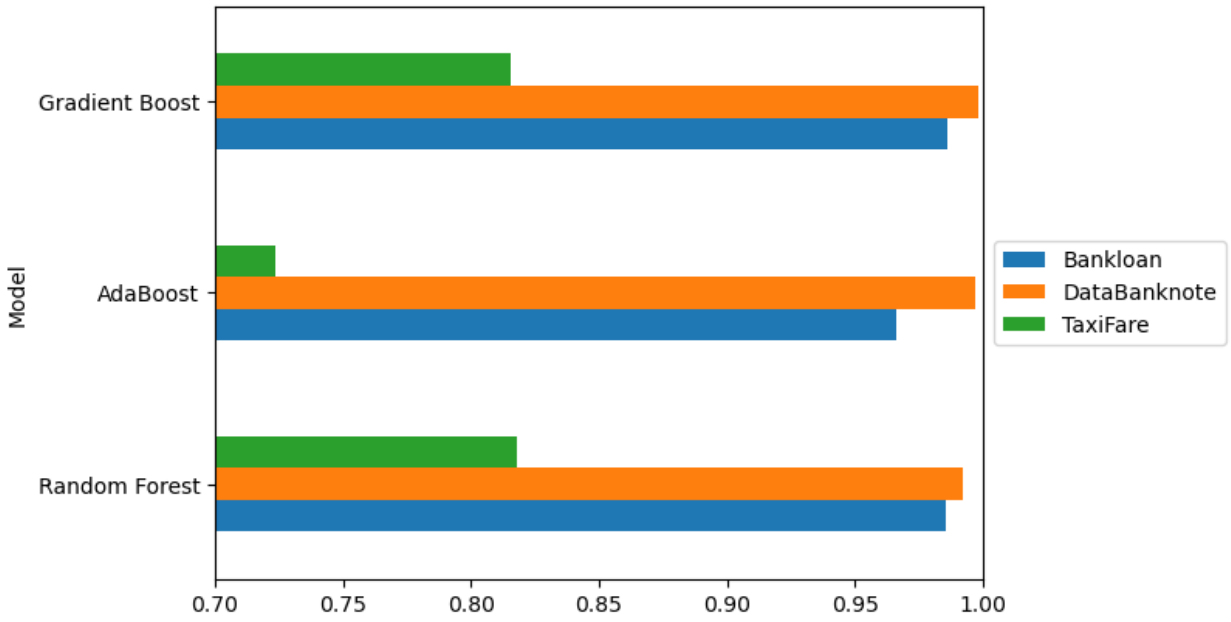
The optimal configuration for the GradientBoostingRegressor on the third dataset involves a learning rate of 0.1, max depth of 4, minimum samples per leaf at 2, minimum samples per split at 2, and 100 estimators. This results in a highly effective regressor, achieving a score of 0.8155 on the training data.

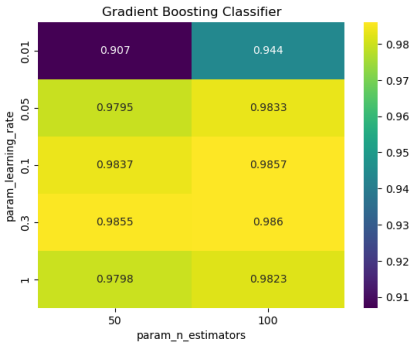
Results – Evaluation of the Model (on the test data)

```
Mean Squared Error: 12.786441048020755
R-squared: 0.8529606527383429
Mean Absolute Error: 1.6151641794733644
```

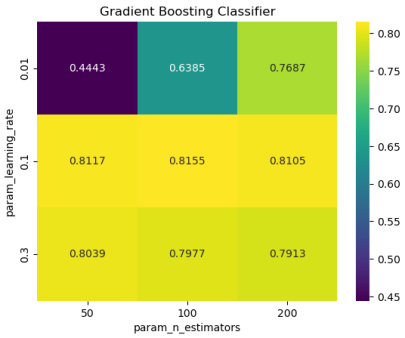
Comparison Table

	Model	Bankloan	DataBanknote	TaxiFare
0	Random Forest	0.985499	0.992191	0.818037
1	AdaBoost	0.966251	0.997070	0.723737
2	Gradient Boost	0.986000	0.998048	0.815500

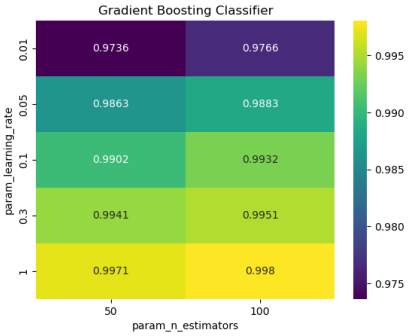




Best Model for Dataset 1
-> Gradient Boost



Best Model for Dataset 2
-> Gradient Boost



Best Model for Dataset 3
-> Gradient Boost

Insights And Conclusions

Bankloan Dataset

Model Performance:

Random Forest: Achieved high accuracy (0.985499).

AdaBoost: High accuracy (0.966251).

Gradient Boost: High accuracy (0.986000).

Reasons:

Strengths: All three models are well-suited for classification tasks where the goal is to predict binary outcomes (0 or 1).

Dataset Characteristics: The Bankloan dataset likely has well-defined patterns that these models can exploit for accurate predictions.

DataBanknote Dataset

Model Performance:

Random Forest: Very high accuracy (0.992191).

AdaBoost: Very high accuracy (0.997070).

Gradient Boost: Excellent accuracy (0.998048).

Reasons:

Strengths: Ensemble methods excel in handling complex patterns and relationships in the data.

Dataset Characteristics: The DataBanknote dataset may have distinct and separable features, allowing ensemble models to achieve high accuracy.

TaxiFare Dataset

Model Performance:

Random Forest: Moderate accuracy (0.818037).

AdaBoost: Lower accuracy (0.723737).

Gradient Boost: Moderate accuracy (0.815500).

Reasons:

Challenges: The TaxiFare dataset involves regression, predicting a continuous numeric value (fare amount) rather than a binary outcome.

Weaknesses: Ensemble models might not perform as well in regression tasks compared to classification tasks.

Dataset Characteristics: The regression nature of the task and the need to predict precise numeric values could pose challenges. The random sample from the large dataset might introduce some variability.

Model Adaptability: Gradient Boost, despite being a boosting algorithm often used for regression, still faces challenges in the regression task. It might require more nuanced hyperparameter tuning for improved performance.

Data Size: The decision to take a random sample from the TaxiFare dataset might introduce variability. The larger dataset might provide more diverse patterns for the models to learn.