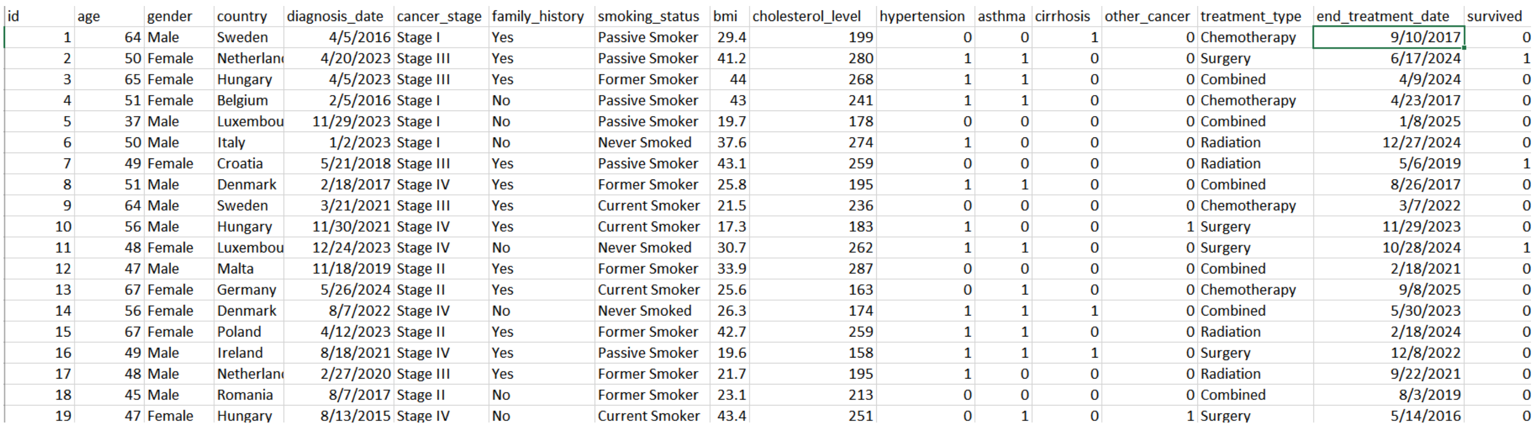
Introduction: Our selected dataset has 17 columns and 890000 rows in total. Here I have selected **"Survived"** as a target column. If a patient has survived it is represented by 1 and if he did not survive it is represented by 0.



id	age	gender	country	diagnosis_date	cancer_stage	family_history	smoking_status	bmi	cholesterol_level	hypertension	asthma	cirrhosis	other_cancer	treatment_type	end_treatment_date	survived
1	64	Male	Sweden	4/5/2016	Stage I	Yes	Passive Smoker	29.4	199	0	0	1	0	Chemotherapy	9/10/2017	0
2	50	Female	Netherlan	4/20/2023	Stage III	Yes	Passive Smoker	41.2	280	1	1	0	0	Surgery	6/17/2024	1
3	65	Female	Hungary	4/5/2023	Stage III	Yes	Former Smoker	44	268	1	1	0	0	Combined	4/9/2024	0
4	51	Female	Belgium	2/5/2016	Stage I	No	Passive Smoker	43	241	1	1	0	0	Chemotherapy	4/23/2017	0
5	37	Male	Luxembou	11/29/2023	Stage I	No	Passive Smoker	19.7	178	0	0	0	0	Combined	1/8/2025	0
6	50	Male	Italy	1/2/2023	Stage I	No	Never Smoked	37.6	274	1	0	0	0	Radiation	12/27/2024	0
7	49	Female	Croatia	5/21/2018	Stage III	Yes	Passive Smoker	43.1	259	0	0	0	0	Radiation	5/6/2019	1
8	51	Male	Denmark	2/18/2017	Stage IV	Yes	Former Smoker	25.8	195	1	1	0	0	Combined	8/26/2017	0
9	64	Male	Sweden	3/21/2021	Stage III	Yes	Current Smoker	21.5	236	0	0	0	0	Chemotherapy	3/7/2022	0
10	56	Male	Hungary	11/30/2021	Stage IV	Yes	Current Smoker	17.3	183	1	0	0	1	Surgery	11/29/2023	0
11	48	Female	Luxembou	12/24/2023	Stage IV	No	Never Smoked	30.7	262	1	1	0	0	Surgery	10/28/2024	1
12	47	Male	Malta	11/18/2019	Stage II	Yes	Former Smoker	33.9	287	0	0	0	0	Combined	2/18/2021	0
13	67	Female	Germany	5/26/2024	Stage II	Yes	Current Smoker	25.6	163	0	1	0	0	Chemotherapy	9/8/2025	0
14	56	Female	Denmark	8/7/2022	Stage IV	No	Never Smoked	26.3	174	1	1	1	0	Combined	5/30/2023	0
15	67	Female	Poland	4/12/2023	Stage II	Yes	Former Smoker	42.7	259	1	1	0	0	Radiation	2/18/2024	0
16	49	Male	Ireland	8/18/2021	Stage IV	Yes	Passive Smoker	19.6	158	1	1	1	0	Surgery	12/8/2022	0
17	48	Male	Netherlan	2/27/2020	Stage III	Yes	Former Smoker	21.7	195	1	0	0	0	Radiation	9/22/2021	0
18	45	Male	Romania	8/7/2017	Stage II	No	Former Smoker	23.1	213	0	0	0	0	Combined	8/3/2019	0
19	47	Female	Hungary	8/13/2015	Stage IV	No	Current Smoker	43.4	251	0	1	0	1	Surgery	5/14/2016	0

Figure-01: A snapshot of our dataset

Class Distribution: Class distribution refers to how the instances of different target labels (classes) are distributed in a dataset. Visualizing this helps identify if the data is balanced or imbalanced. A bar plot is commonly used to show the number of samples in each class.

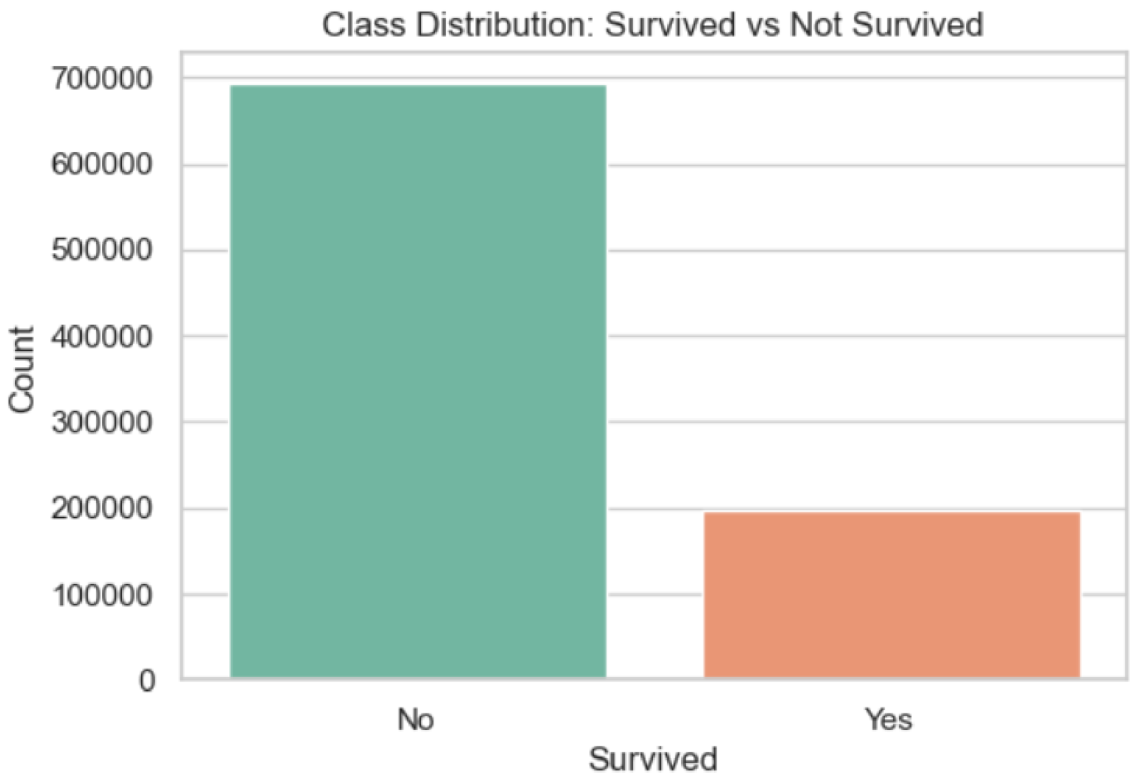


Figure-02: Distribution of target class

Histogram: A histogram is a graphical representation that shows the distribution of a numerical feature. It divides the data into intervals (called bins) and plots the frequency (count) of data points in each bin. Histograms help identify:

- Skewness of data (left, right, or symmetric)
- Presence of outliers
- Spread and central tendency
- Useful for feature understanding and preprocessing decisions (like scaling or normalization).

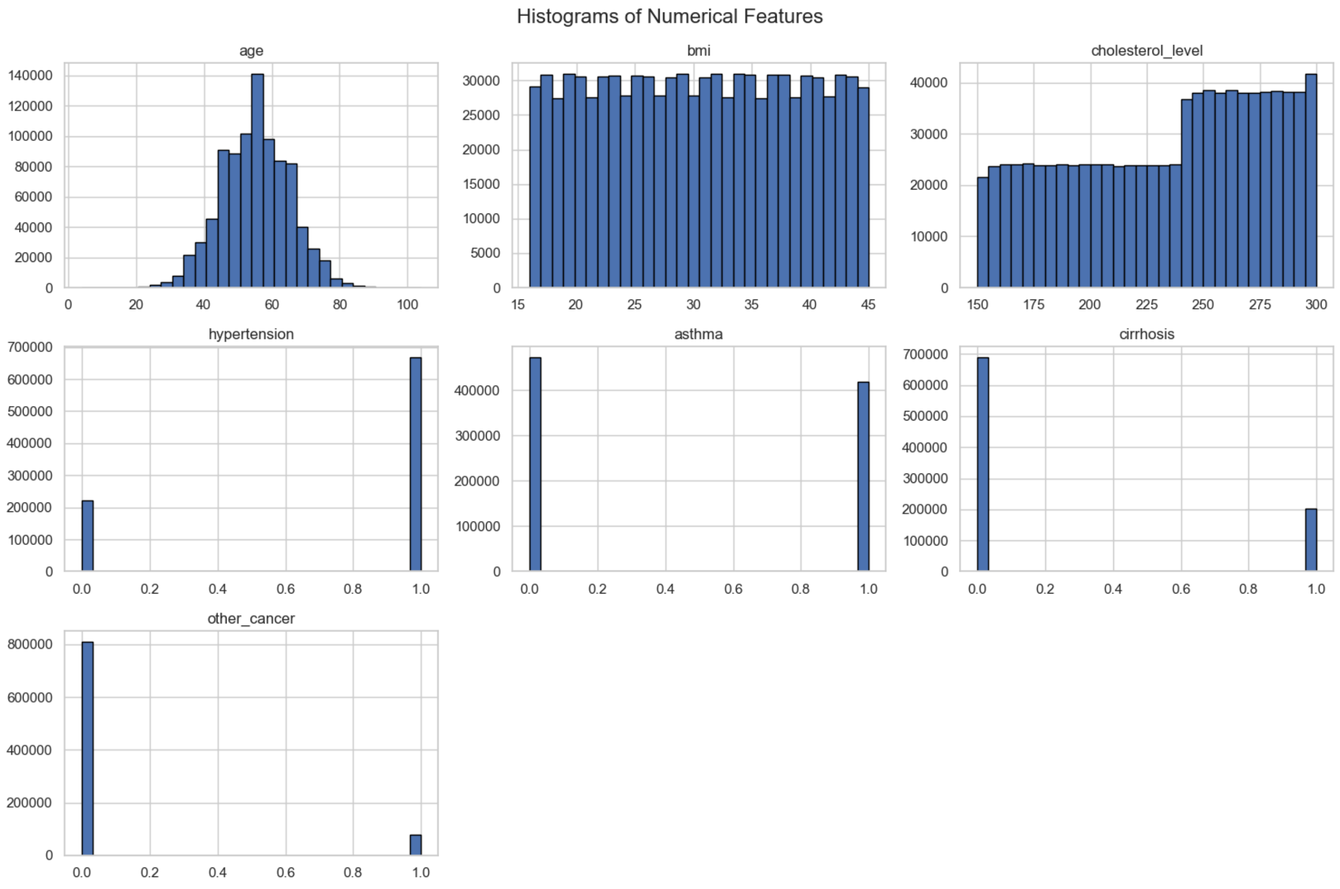


Figure-03: Histograms of numerical features

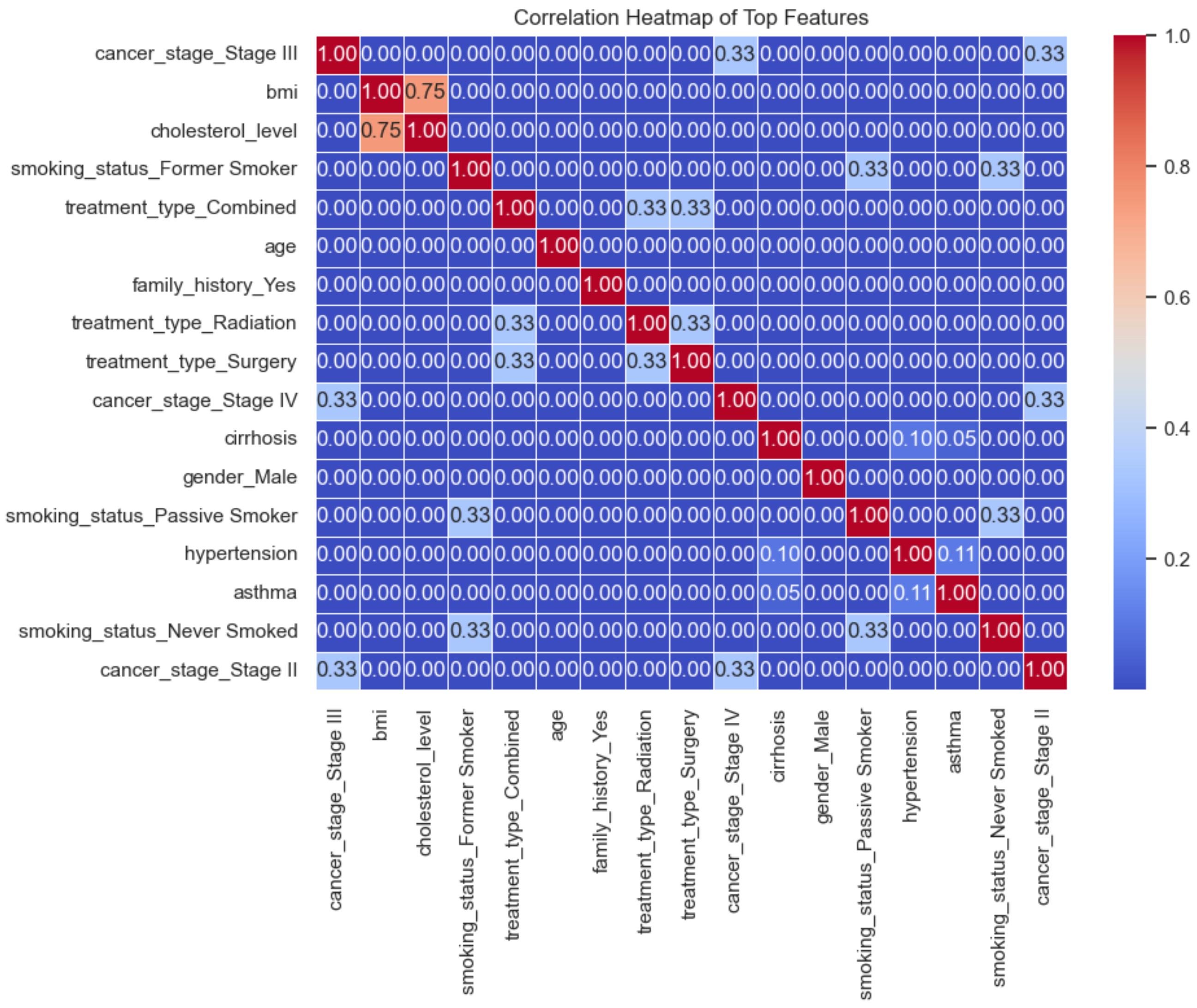
Here, I have dropped the column **id** and **survived** and divided each numeric features in 30 intervals.

Correlation Heatmap: A correlation heatmap is a visual tool used to display the correlation coefficients between numerical features in a dataset. Correlation values range from -1 to +1:

- +1 indicates Perfect positive correlation
- -1 indicates Perfect negative correlation
- 0 indicates No linear correlation

The heatmap helps to:

- Identify strong relationships between features
- Detect redundant features (highly correlated ones)
- Guide feature selection to avoid multicollinearity



	cancer_stage_Stage III	bmi	cholesterol_level	smoking_status_Former Smoker	treatment_type_Combined	age	family_history_Yes	treatment_type_Radiation	treatment_type_Surgery	cancer_stage_Stage IV	cirrhosis	gender_Male	smoking_status_Passive Smoker	hypertension	asthma	smoking_status_Never Smoked	cancer_stage_Stage II
cancer_stage_Stage III	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.33
bmi	0.00	1.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
cholesterol_level	0.00	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
smoking_status_Former Smoker	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.33	0.00
treatment_type_Combined	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.33	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
age	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
family_history_Yes	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
treatment_type_Radiation	0.00	0.00	0.00	0.00	0.33	0.00	0.00	1.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
treatment_type_Surgery	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.33	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
cancer_stage_Stage IV	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
cirrhosis	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.10	0.05	0.00	0.00
gender_Male	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
smoking_status_Passive Smoker	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.33	0.00
hypertension	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	1.00	0.11	0.00	0.00
asthma	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.11	1.00	0.00	0.00
smoking_status_Never Smoked	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	1.00	0.00
cancer_stage_Stage II	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure-04: Correlation heatmap of top features

First, I dropped the irrelevant columns such as- **'id'**, **'diagnosis_date'**, **'end_treatment_date'**, **'country'**, **'survived'** and then dropped high cardinality columns by dropping categorical features with too many unique values(>30). Then, I converted categorical variables into binary dummy variables for One-hot coding.

After that, I have divided the data into testing set (20%) and training set (80%). I have trained a Random Forest Model and extracted feature importances followed by sorting the features by importances.

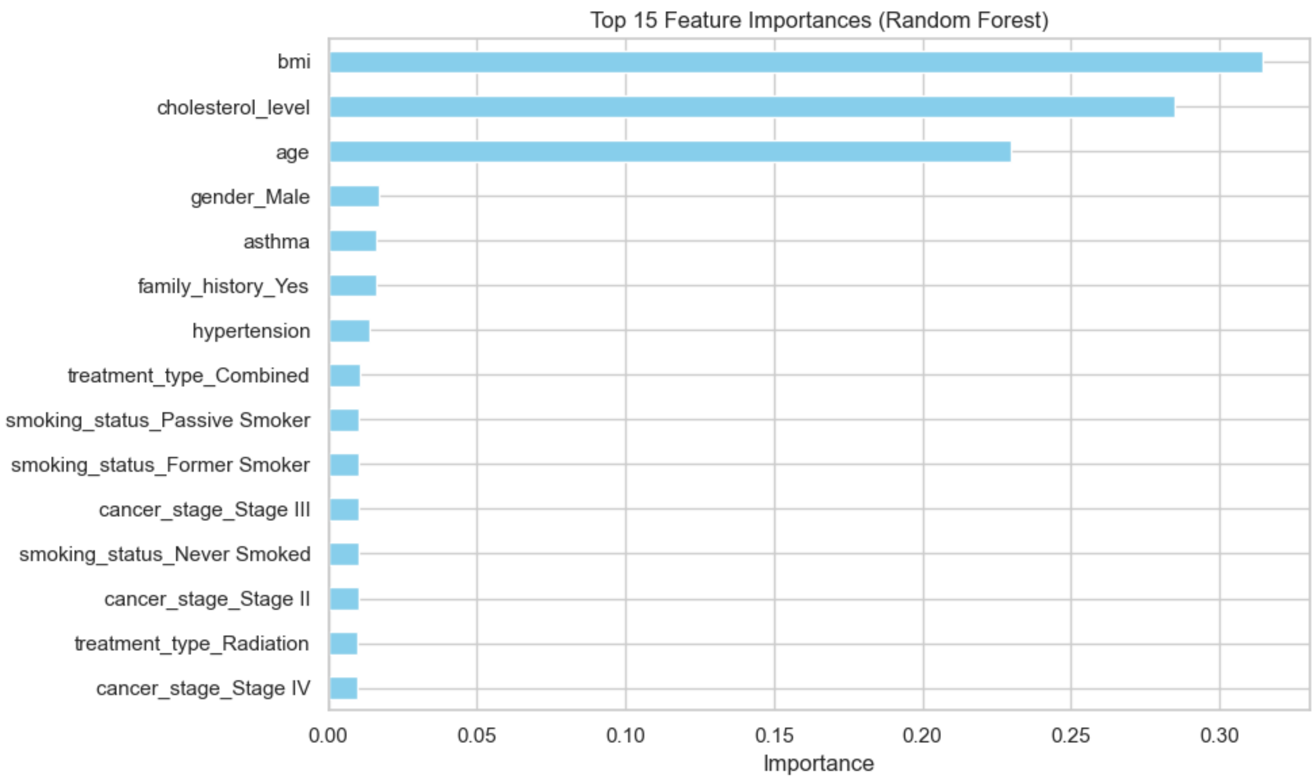
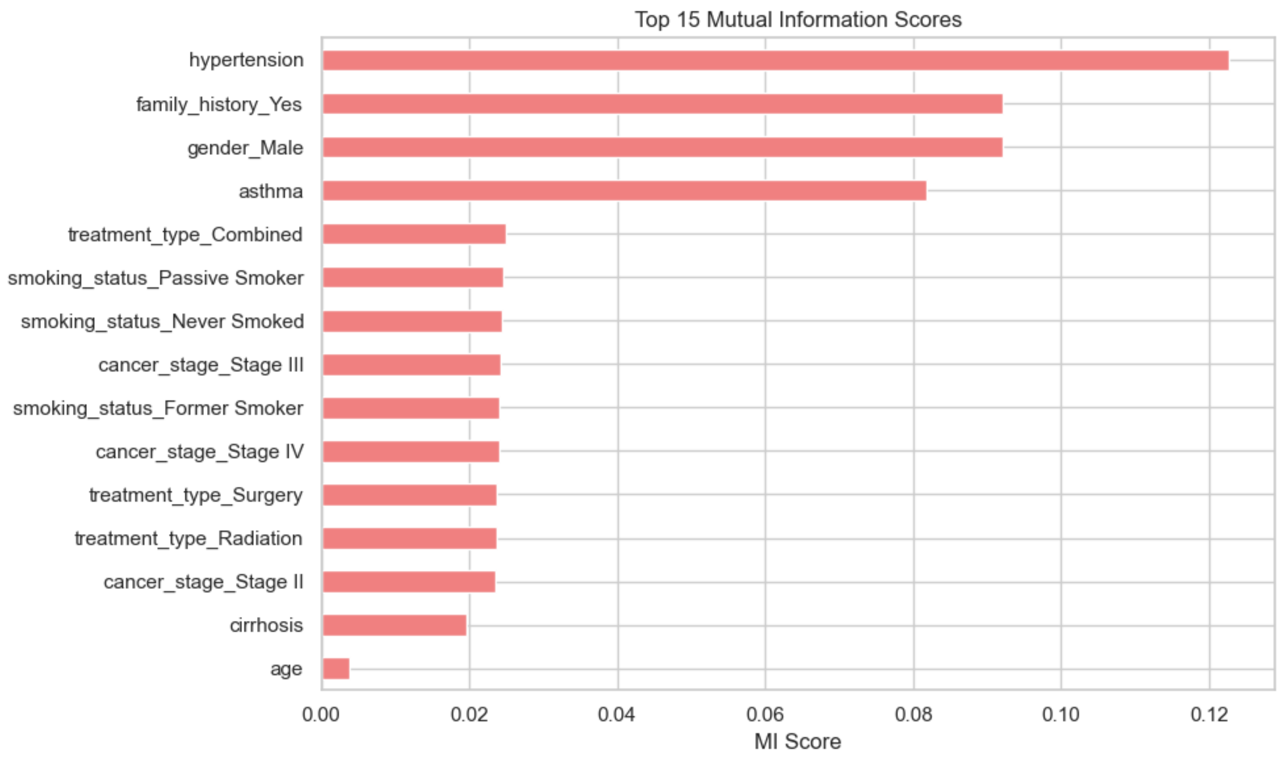
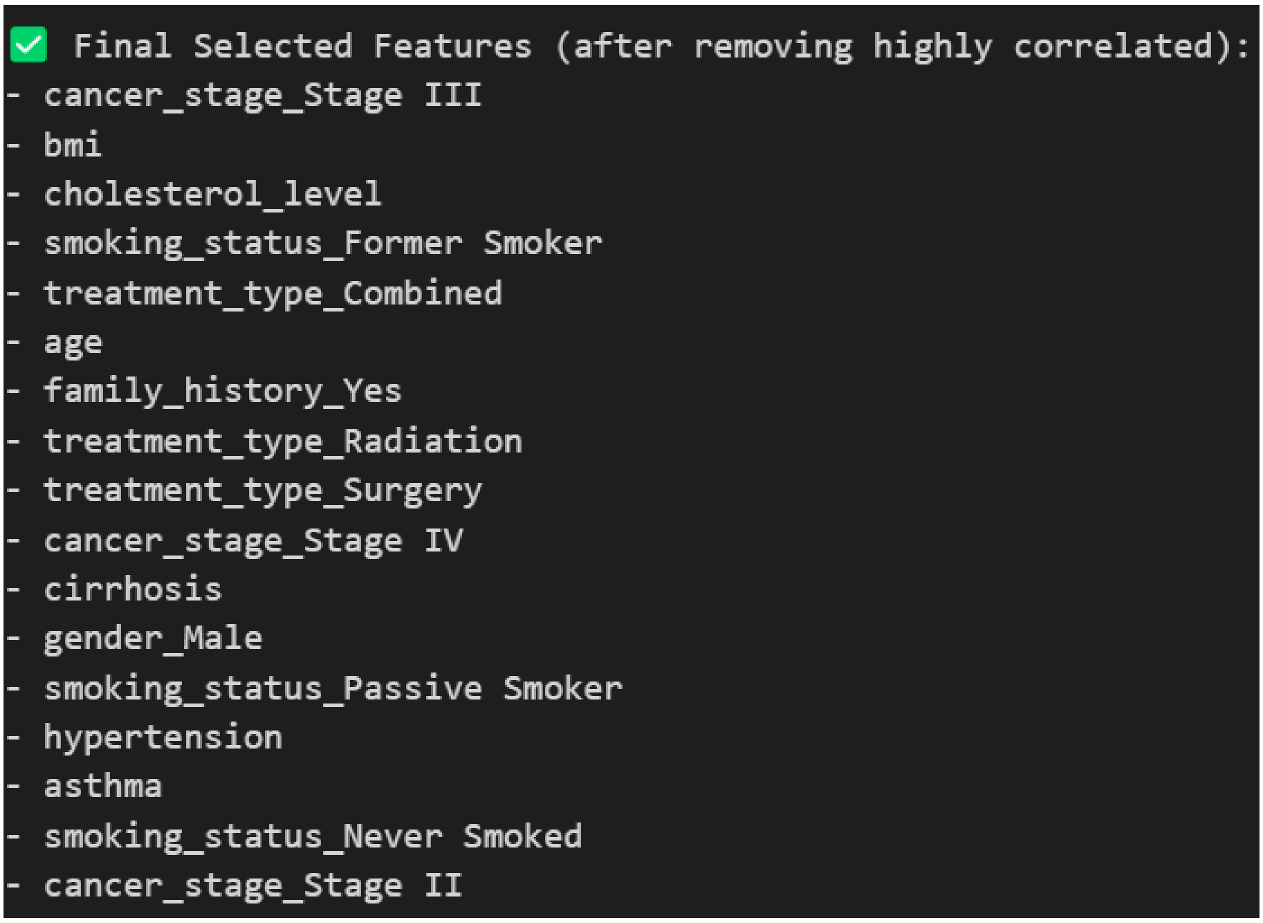


Figure-05: Top 15 feature importances of random forest

Similarly, I have figured out feature importance by Mutual Information. It is a measure of dependency between each feature and the target.



Then I have merged the top features from both Random Forest and Mutual Information method to create a comprehensive feature. Finally, I have created the correlation heatmap of top features and removed highly correlated features. Then I have printed the final features.

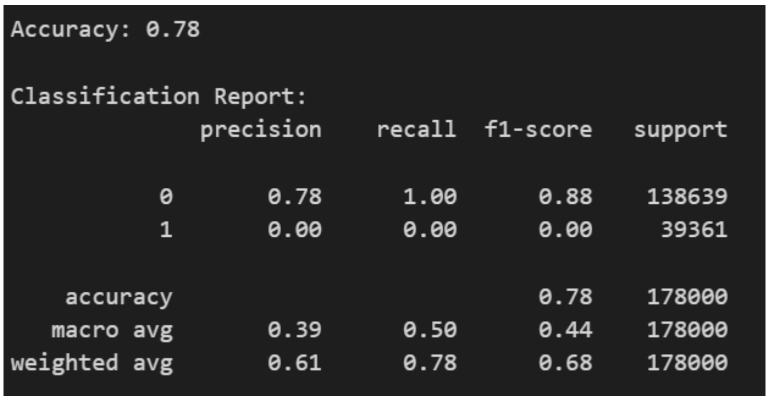


```
✅ Final Selected Features (after removing highly correlated):
- cancer_stage_Stage III
- bmi
- cholesterol_level
- smoking_status_Former Smoker
- treatment_type_Combined
- age
- family_history_Yes
- treatment_type_Radiation
- treatment_type_Surgery
- cancer_stage_Stage IV
- cirrhosis
- gender_Male
- smoking_status_Passive Smoker
- hypertension
- asthma
- smoking_status_Never Smoked
- cancer_stage_Stage II
```

Figure-06: List of Final Selected Features

Logistic Regression: Logistic Regression is a supervised classification algorithm used to predict the probability of a categorical outcome, typically binary (e.g., 0 = No, 1 = Yes).

- **Output**: Probability between 0 and 1
- **Decision Rule**: Classify as 1 if probability ≥ 0.5, otherwise 0
- **Best for**: Binary classification problems



```
Accuracy: 0.78

Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.88    138639
           1       0.00      0.00      0.00     39361

    accuracy                           0.78    178000
   macro avg       0.39      0.50      0.44    178000
weighted avg       0.61      0.78      0.68    178000
```

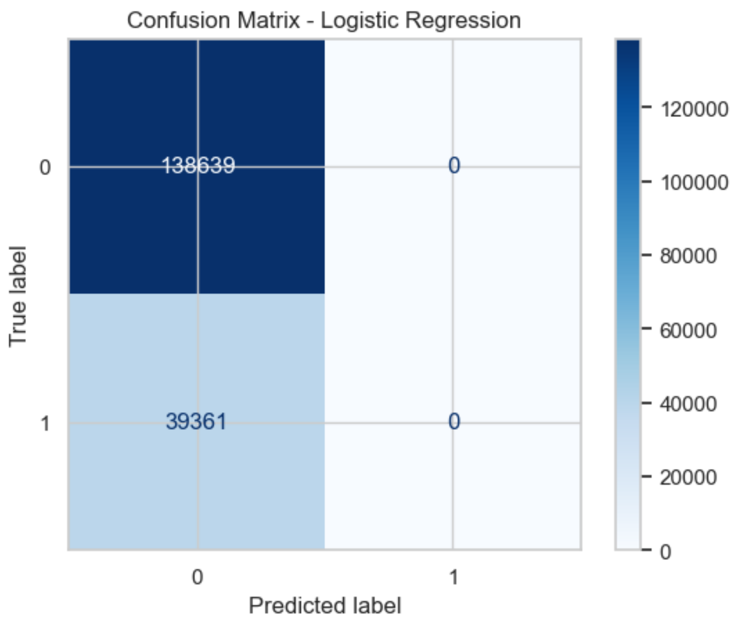
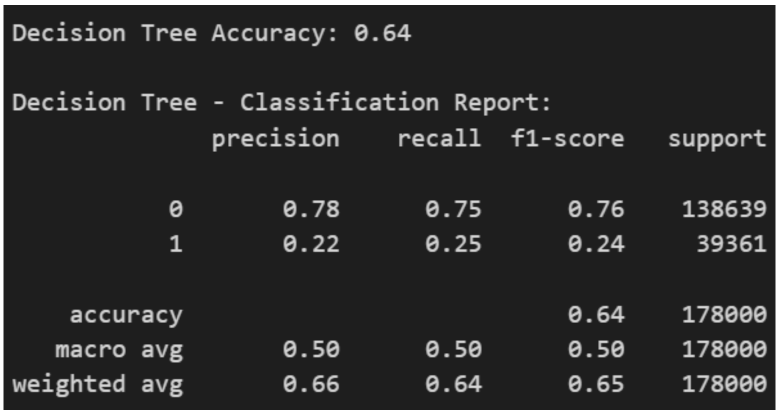


Figure-07: Accuracy, classification report and confusion matrix of Logistic Regression model

In the Logistic Regression model, I have used max_iter = 1000 and solver = ‘liblinear’. Here, increased iterator helps to ensure convergence.

Decision Tree: A Decision Tree is a supervised learning algorithm used for classification and regression. It splits data into branches based on feature values, forming a tree-like structure of decisions.

- Internal nodes: Conditions based on features
- Leaves: Final class labels or predictions
- Splits aim to maximize information gain or reduce impurity (e.g., Gini, entropy)
- Easy to interpret and visualize
- Prone to overfitting if not pruned



```
Decision Tree Accuracy: 0.64

Decision Tree - Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.75      0.76    138639
           1       0.22      0.25      0.24     39361

    accuracy                           0.64    178000
   macro avg       0.50      0.50      0.50    178000
weighted avg       0.66      0.64      0.65    178000
```

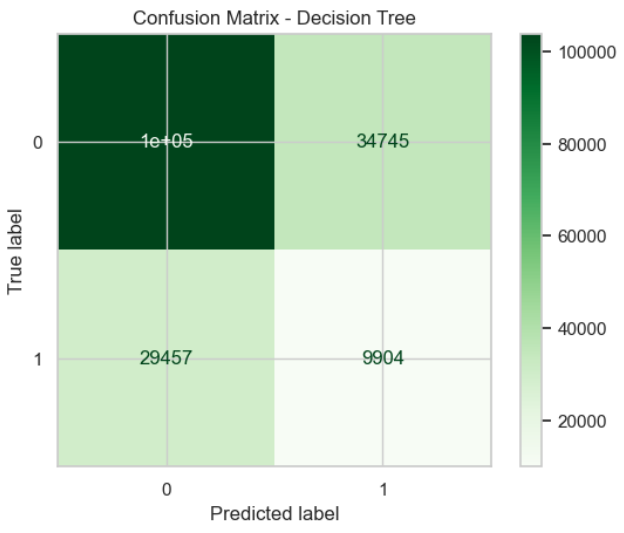
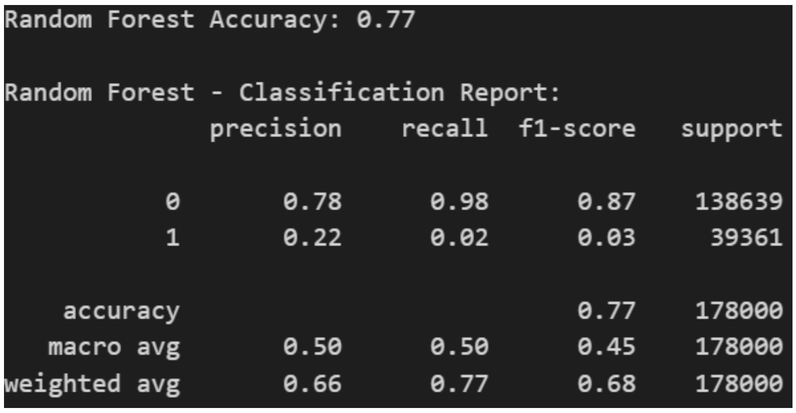


Figure-09: Accuracy, classification report and confusion matrix of Decision Tree model

In the Decision tree model, I have declared random_state = 42 for consistent results.

Random Forest: A Random Forest is an ensemble machine learning model that builds multiple decision trees and combines their outputs to improve prediction accuracy and reduce overfitting.

- Each tree is trained on a random subset of the data (bagging).
- Predictions are made by majority vote (for classification).
- It is more robust and accurate than a single decision tree.
- Works well with high-dimensional data and can estimate feature importance.



```
Random Forest Accuracy: 0.77

Random Forest - Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.98      0.87    138639
           1       0.22      0.02      0.03     39361

    accuracy                           0.77    178000
   macro avg       0.50      0.50      0.45    178000
weighted avg       0.66      0.77      0.68    178000
```

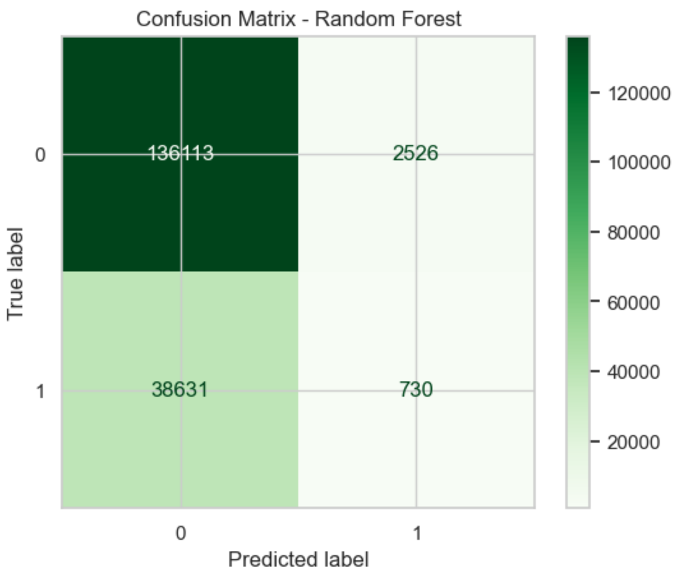


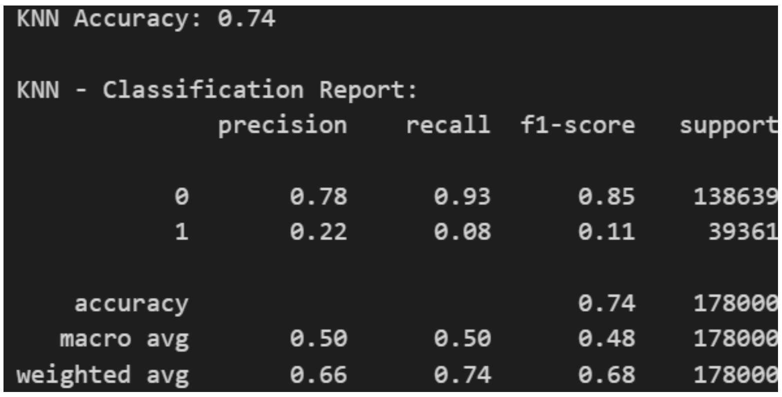
Figure-10: Accuracy, classification report and confusion matrix of Random Forest model

In this model, I have user n_estimators = 100, which builds 100 trees and random_state = 42 which ensures reproducibility.

K-Nearest Neighbors: K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression. It is a lazy learner, meaning it does not learn an explicit model during training but makes decisions during prediction.

- For classification, it assigns a class based on the majority class among the 'k' nearest neighbors.
- Distance (usually Euclidean) is used to find the neighbors.

- Sensitive to the value of k and the scale of the data.
- Performs best with clean and well-scaled data.
-



```
KNN Accuracy: 0.74

KNN - Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.93      0.85    138639
           1       0.22      0.08      0.11     39361

    accuracy                           0.74    178000
   macro avg       0.50      0.50      0.48    178000
weighted avg       0.66      0.74      0.68    178000
```

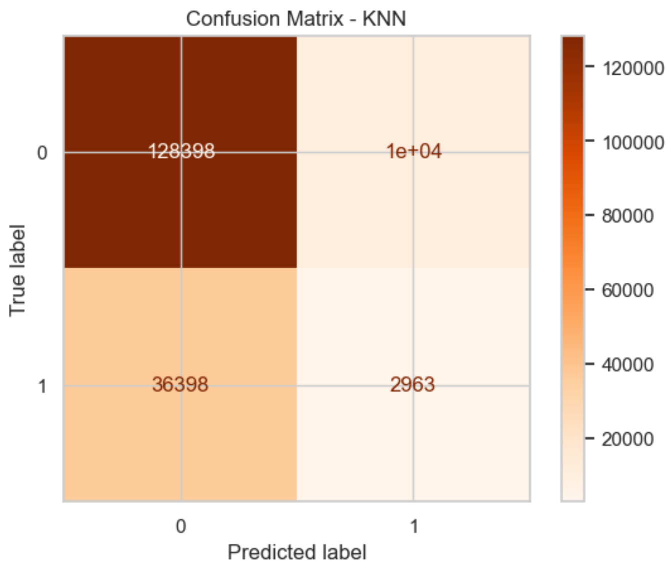
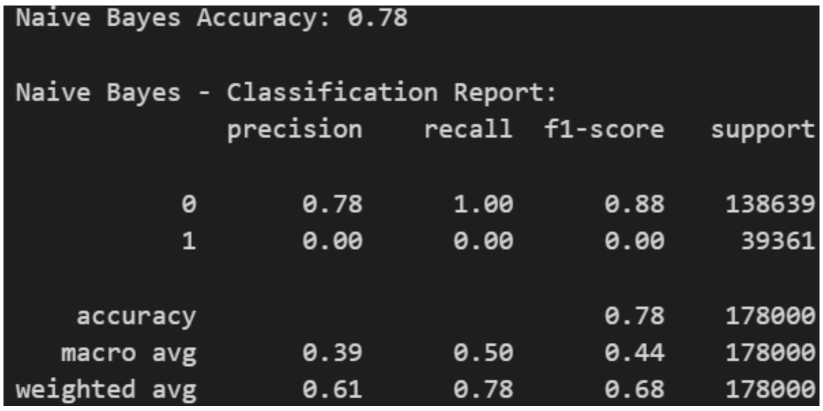


Figure- 11: Accuracy, classification report and confusion matrix of K-Nearest Neighbor model

In this model, I have set n_neighbors = 5. It indicates that the 5 nearest neighbors are considered while making guesses.

Naive Bayes: Naive Bayes is a probabilistic classifier based on Bayes' Theorem, with a strong (naive) assumption that all features are independent of each other given the class. Variants of Naive Bayes:

- GaussianNB– assumes normal distribution of features (used for continuous data).
- MultinomialNB– used for count data (e.g. text classification).
- BernoulliNB– used for binary / boolean features.



```
Naive Bayes Accuracy: 0.78

Naive Bayes - Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.88    138639
           1       0.00      0.00      0.00     39361

    accuracy                           0.78    178000
   macro avg       0.39      0.50      0.44    178000
weighted avg       0.61      0.78      0.68    178000
```

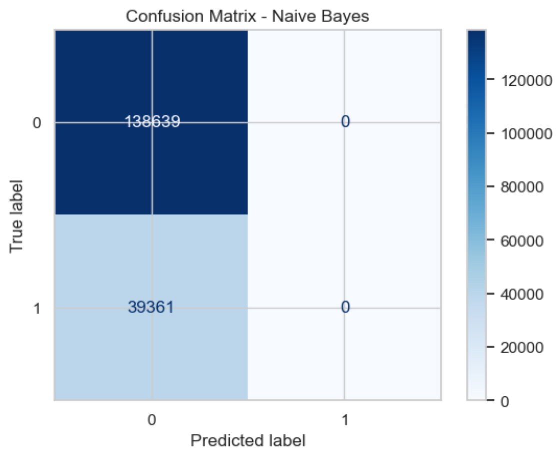
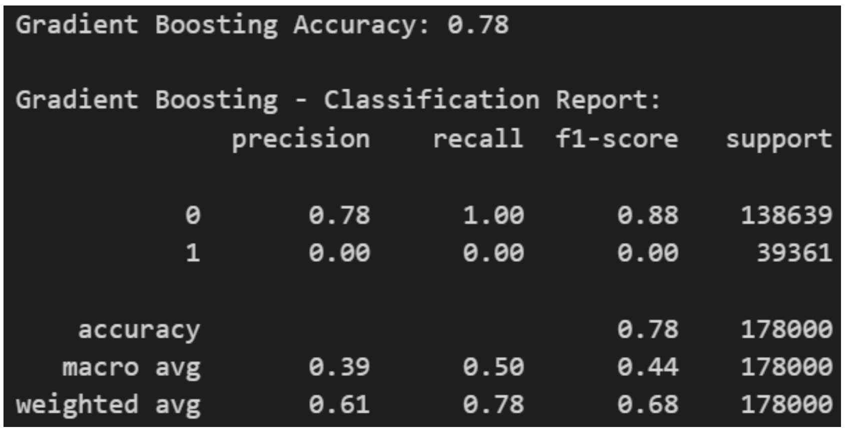


Figure-12: Accuracy, classification report and confusion matrix of Naive Bayes model

Here, I have used Gaussian Naïve Bayes model.

Gradient Boosting Classifier: Gradient Boosting is a powerful ensemble machine learning algorithm used for both classification and regression tasks. It builds models sequentially, where each new model corrects the errors made by the previous ones.
Key Concepts:

- Uses decision trees (typically shallow) as weak learners.
- Optimizes a loss function using gradient descent techniques.
- Very effective in capturing complex patterns and non-linear relationships.
- Tends to perform better than simpler models but can overfit if not tuned properly.



```
Gradient Boosting Accuracy: 0.78

Gradient Boosting - Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.88    138639
           1       0.00      0.00      0.00     39361

    accuracy                           0.78    178000
   macro avg       0.39      0.50      0.44    178000
weighted avg       0.61      0.78      0.68    178000
```

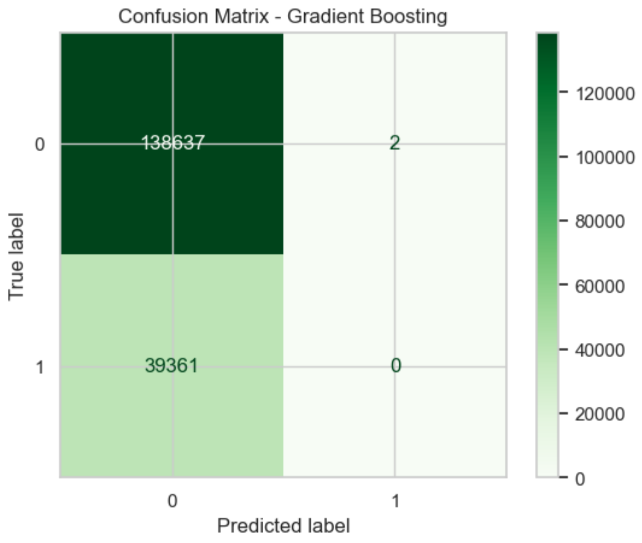


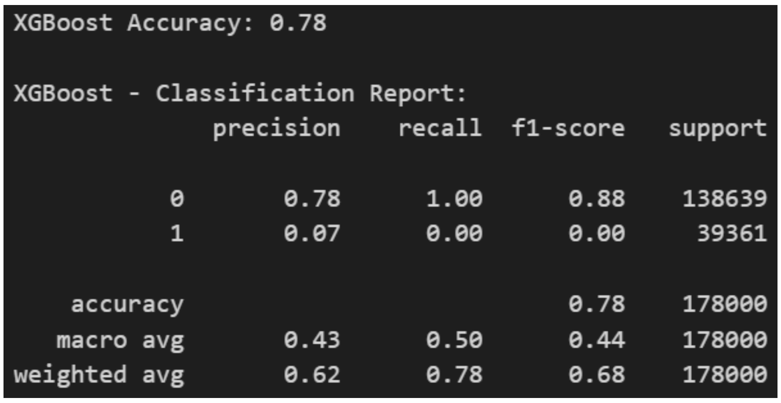
Figure-13: Accuracy, classification report and confusion matrix of Gradient Boosting model

For this Gradient boosting model, I have used n_estimator = 100, learning_rate = 0.1 and random_state = 42. Here, the model builds 100 boosting stages, and contribution of each tree is 0.1 and random state 42 ensures reproducibility.

XGBoosting: XGBoost (Extreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm designed for speed and performance. It is widely used in machine learning competitions and real-world applications due to its accuracy and efficiency.
Key Features:

- Uses boosted decision trees.
- Incorporates regularization (L1 & L2) to prevent overfitting.

- Supports parallel processing and early stopping.
- Handles missing values internally.
- Works well with tabular data.



```
XGBoost Accuracy: 0.78

XGBoost - Classification Report:
             precision    recall  f1-score   support

           0       0.78      1.00      0.88    138639
           1       0.07      0.00      0.00     39361

    accuracy                           0.78    178000
   macro avg       0.43      0.50      0.44    178000
weighted avg       0.62      0.78      0.68    178000
```

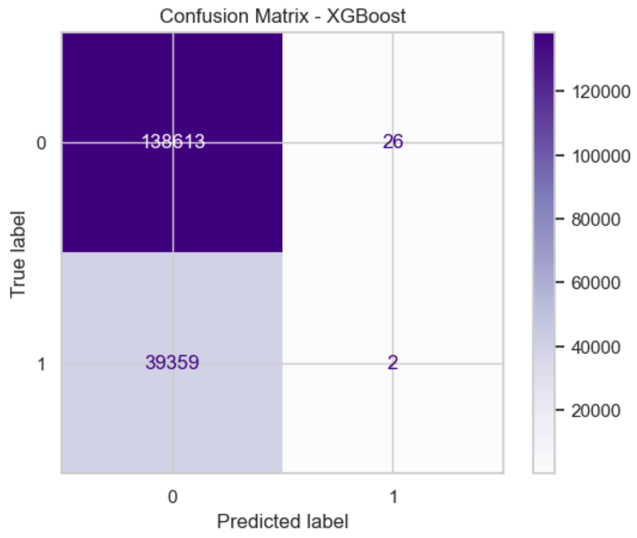


Figure-14: Accuracy, classification report and confusion matrix of XGBoost model

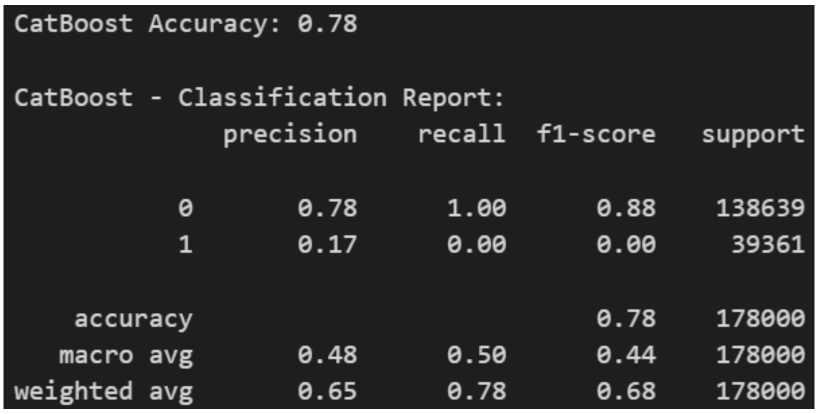
In this XGBoost Classifier model, I have used eval_metric = logloss. Log Loss, also known as **logarithmic loss** or **cross-entropy loss**, is a performance metric used to evaluate the accuracy of a classification model where the prediction is a probability between 0 and 1.
It penalizes false predictions with high confidence more than those made with less confidence.
Benefits of using logloss:

- Suitable for **binary classification**.
- Encourages models to output **probabilistic predictions**, not just hard class labels.
- Gives **higher penalty** for confident wrong predictions (e.g., predicting 0.99 when the true label is 0).

CatBoost Model: CatBoost is a high-performance, open-source gradient boosting algorithm developed by Yandex, specifically optimized for categorical features.
Key Features:

- Handles categorical data natively — no need for one-hot or label encoding.
- Built-in support for missing values.
- Fast training and robust performance, especially on tabular datasets.
- Resistant to overfitting due to techniques like ordered boosting.

CatBoost is widely used for classification, regression, and ranking tasks, especially in data with categorical variables.



```
CatBoost Accuracy: 0.78

CatBoost - Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.88    138639
           1       0.17      0.00      0.00     39361

    accuracy                           0.78    178000
   macro avg       0.48      0.50      0.44    178000
weighted avg       0.65      0.78      0.68    178000
```

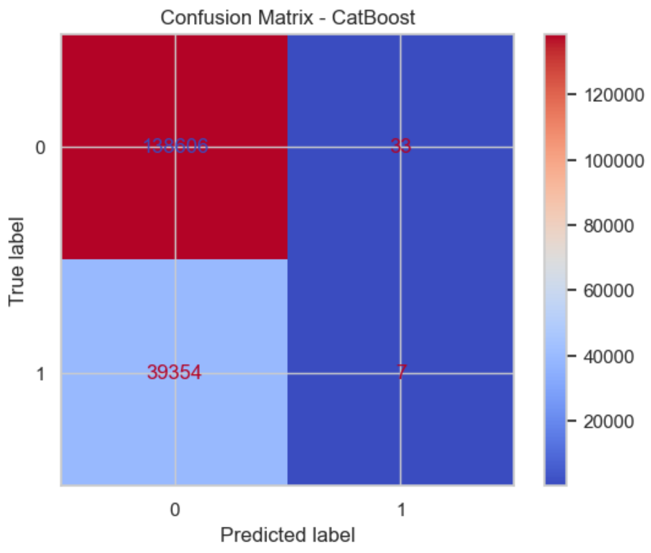


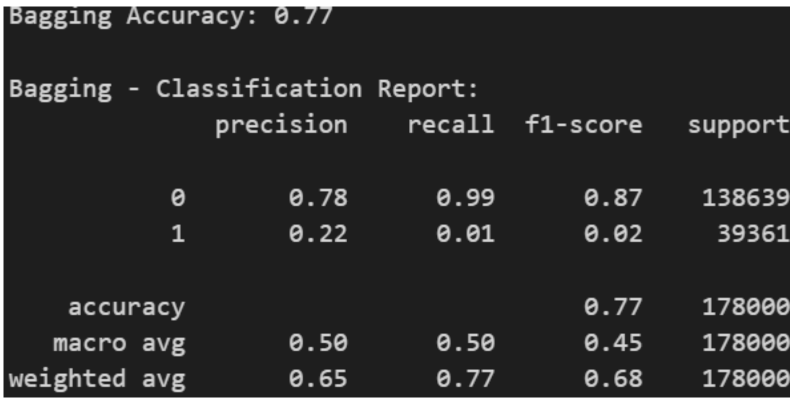
Figure-15: Accuracy, classification report and confusion matrix of CatBoost model

In this CatBoost Model, I have set verbose = 0 and random_state = 42. Here, verbose = 0 is just for suppressing output. It does not affect the overall performance of the model.

Bagging Classifier: Bagging (Bootstrap Aggregating) is an ensemble learning technique that improves model stability and accuracy by training multiple models on random subsets of the data and combining their outputs.

Key Points:

- Reduces variance and helps avoid overfitting.
- Common base models: Decision Trees, but any classifier can be used.
- Final prediction is usually made by majority voting (for classification).



```
Bagging Accuracy: 0.77

Bagging - Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.99      0.87    138639
           1       0.22      0.01      0.02     39361

    accuracy                           0.77    178000
   macro avg       0.50      0.50      0.45    178000
weighted avg       0.65      0.77      0.68    178000
```

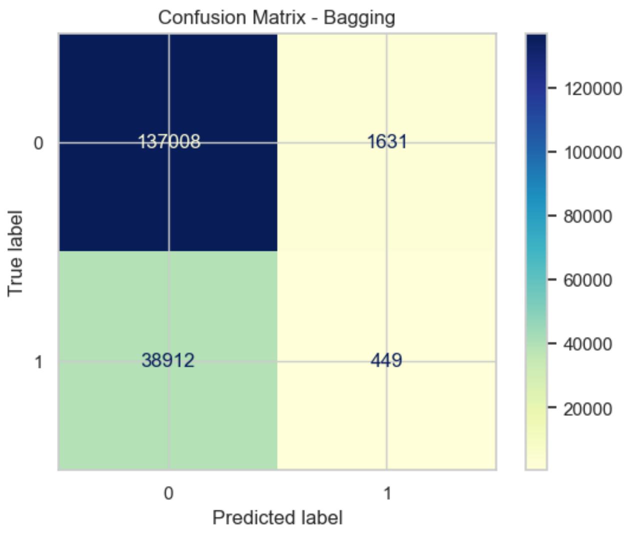


Figure-16: Accuracy, classification report and confusion matrix of Bagging Classifier model

In this bagging classifier model, the base estimator is a Decision Tree Model. n_estimator = 50 builds 50 such models on different bootstrapped samples. The model predicts the outcomes for the test data by aggregating the outputs from all 50 decision trees via majority vote.

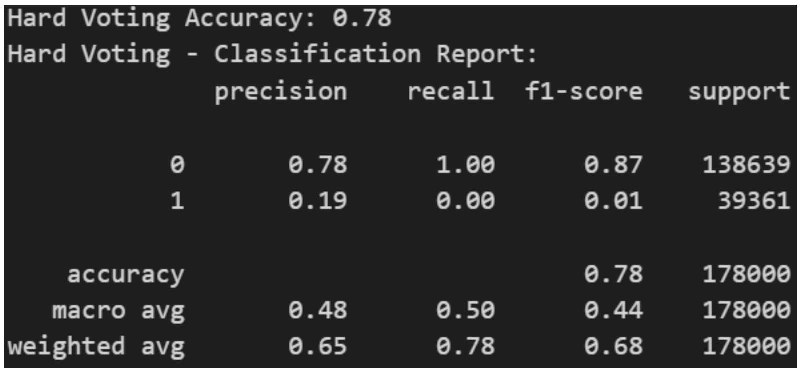
Voting Classifier: A Voting Classifier is an ensemble machine learning model that combines the predictions from multiple different models (e.g., Logistic Regression, KNN, Random Forest) to improve performance.

There are two main types:
Hard Voting

- Takes the majority class label predicted by each base model.
- Final output = mode (most frequent class).
- Use when models don't support probability outputs (predict_proba()).

Soft Voting

- Averages the predicted probabilities from each model.
- Chooses the class with the highest average probability.
- More flexible and often more accurate than hard voting.
- Requires all base models to support predict_proba().



```
Hard Voting Accuracy: 0.78
Hard Voting - Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.87    138639
           1       0.19      0.00      0.01     39361

    accuracy                           0.78    178000
   macro avg       0.48      0.50      0.44    178000
weighted avg       0.65      0.78      0.68    178000
```

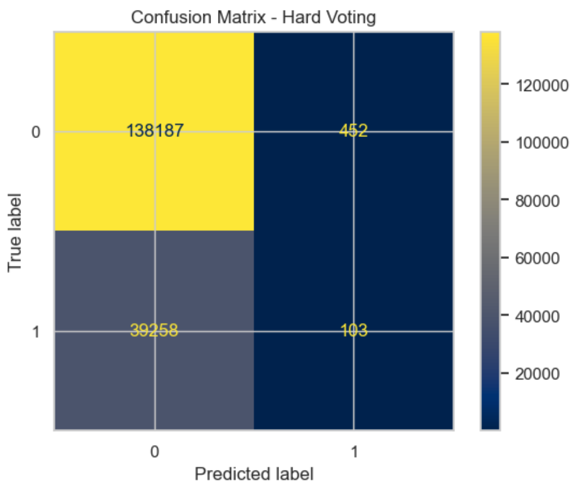
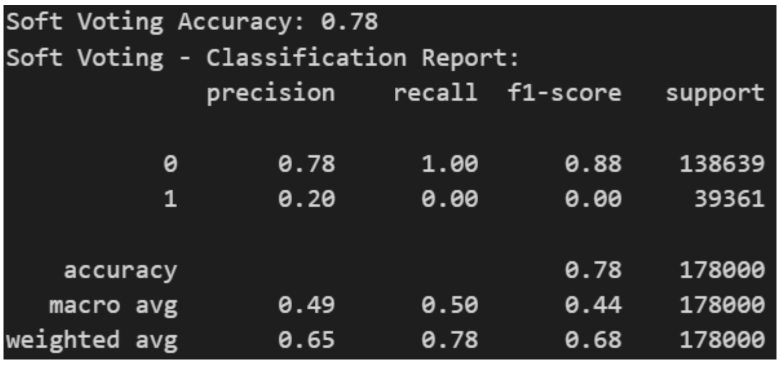


Figure-17: Accuracy, classification report and confusion matrix of Hard Voting model



```
Soft Voting Accuracy: 0.78
Soft Voting - Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.88    138639
           1       0.20      0.00      0.00     39361

    accuracy                           0.78    178000
   macro avg       0.49      0.50      0.44    178000
weighted avg       0.65      0.78      0.68    178000
```

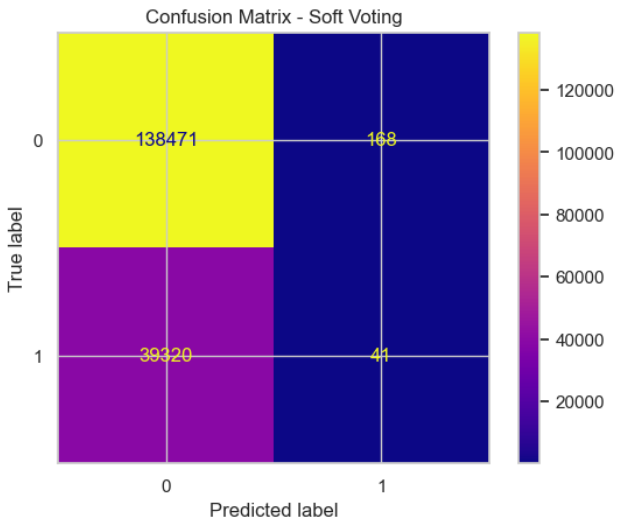


Figure-17: Accuracy, classification report and confusion matrix of Soft Voting model

For this voting classifier model, I have used 3 different classifiers and they are- Logistic Regression, Random Forest and KNeighbors.

Hard voting makes predictions by majority vote. Each model votes for a class and the class with the most votes is selected.

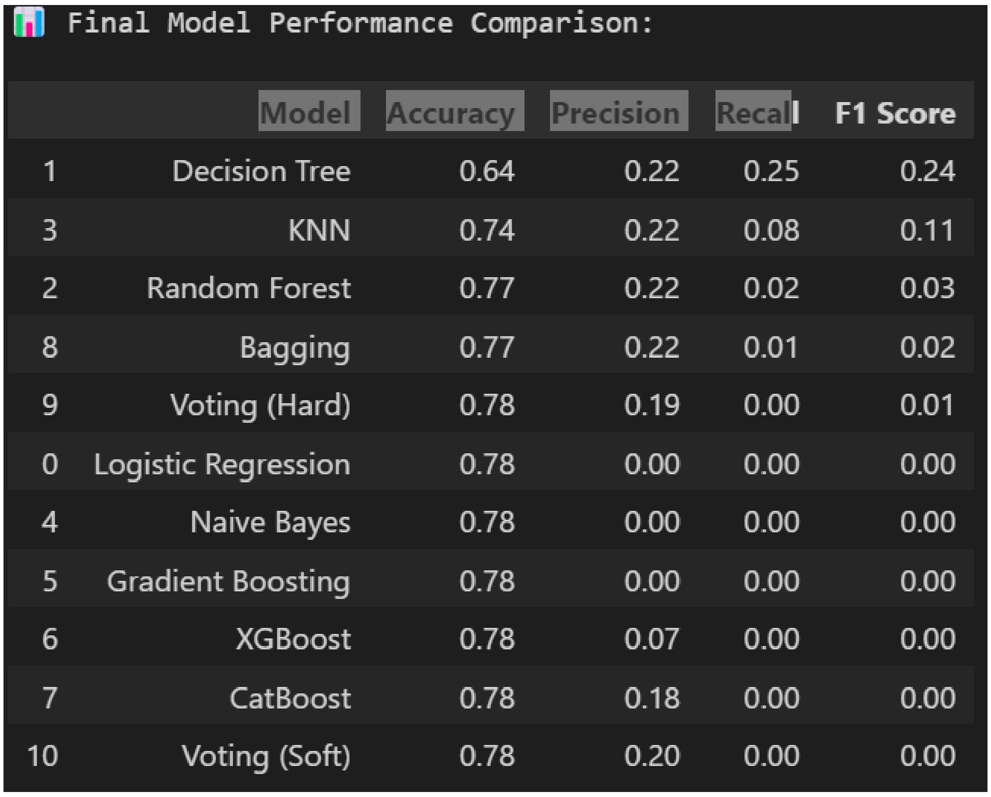
Soft voting uses averaged probabilities from all models. It selects the class with the highest probability rate.

Accuracy: It is the ratio of correctly predicted instances to total instances. It tells how often the model is right and best used when classes are balanced, and all errors are equally important. But it can be misleading in imbalanced datasets.

Precision: It is the proportion of correctly predicted positive instances out of all predicted positives. It focuses on the quality of positive predictions and best used when false positives are costly and dangerous.

Recall: The proportion of actual positives that were actually predicted. It measures how many actual positive values were correctly found. Best used when missing a positive case is risky or costly.

F1 Score: It is the harmonic mean of precise and recall. It gives a single score that balances both precision and recall. This is best used when dataset is imbalanced, we need to balance false positive and false negative values and precision and recall are both critical.



```
📊 Final Model Performance Comparison:
```

	Model	Accuracy	Precision	Recall	F1 Score
1	Decision Tree	0.64	0.22	0.25	0.24
3	KNN	0.74	0.22	0.08	0.11
2	Random Forest	0.77	0.22	0.02	0.03
8	Bagging	0.77	0.22	0.01	0.02
9	Voting (Hard)	0.78	0.19	0.00	0.01
0	Logistic Regression	0.78	0.00	0.00	0.00
4	Naive Bayes	0.78	0.00	0.00	0.00
5	Gradient Boosting	0.78	0.00	0.00	0.00
6	XGBoost	0.78	0.07	0.00	0.00
7	CatBoost	0.78	0.18	0.00	0.00
10	Voting (Soft)	0.78	0.20	0.00	0.00

Figure-18: Model performance comparison table of the performed models

It is observed from the table that models except Decision Tree and KNN all report around 78% accuracy.

Precision, Recall and F1 score are very low for almost all models.

Precision: Mostly around 0.00 to 0.22

Recall: Near zero (0.00 to 0.25)

F1 score: Often 0, indicating the harmonic mean of precision and recall is extremely low.

As the Accuracy is high but other parameters are low, it can be said that the dataset is **Severely Imbalanced.**

High Accuracy: The model correctly predicts the majority class most of the time.

Low Recall / F1 score / Precision: The model fails to identify minority class effectively.

AUC-ROC Curve: AUC-ROC stands for Area Under the Curve – Receiver Operating Characteristics. It is a performance evaluation metric for classification models, especially for binary classification problems.

ROC Curve: Plots True Positive Rate (TPR) vs False Positive Rate (FPR) at different threshold values.

AUC (Area Under Curve): A single number summary of the ROC curve.

AUC = 1.0 indicates Perfect classifier
AUC = 0.5 indicates Random guessing
AUC < 0.5 indicates Worse than random (something is wrong)

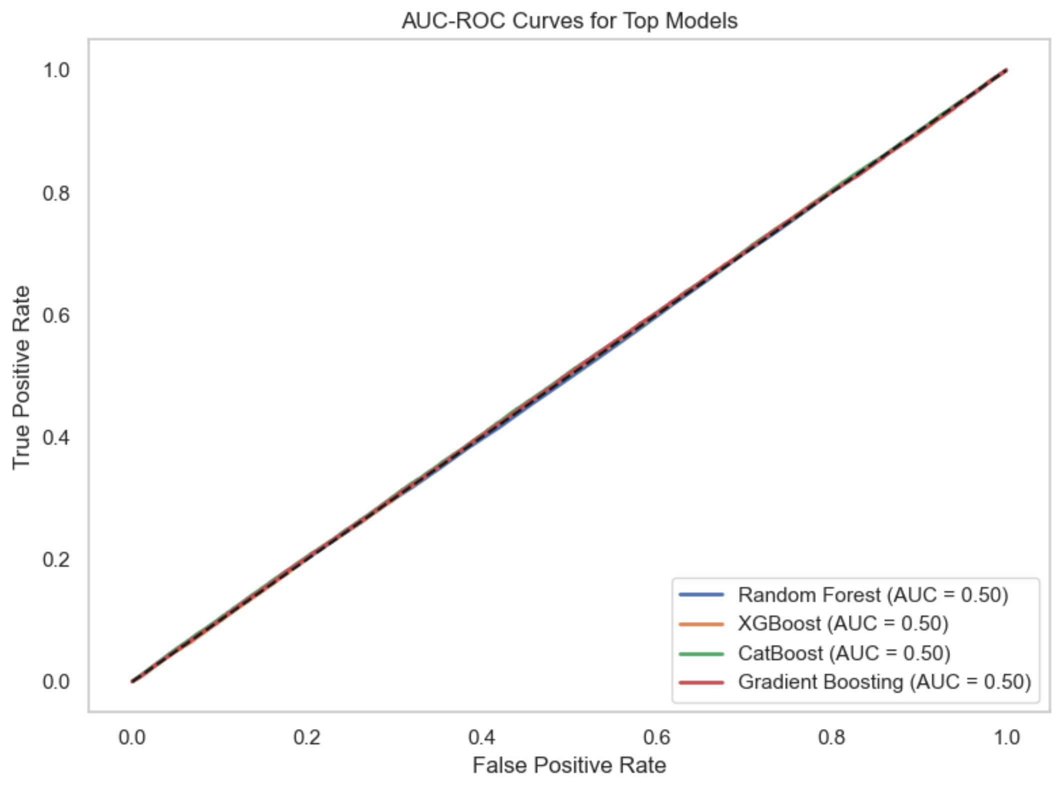


Figure-19: AUC-ROC curve of the top performing models

AUC = 0.5 indicates no discrimination between the positive and negative classes. The models can not distinguish between classes better than random chance.