PROJECT REPORT

DEVELOPED BY:

YENIYAN SHIFA

CONTENTS

Introduction

This report presents an end-to-end machine learning project, covering dataset exploration, preprocessing, feature engineering, modeling, and evaluation. The primary goal of this study is to build predictive models based on the given dataset and optimize them for improved performance.

1. Dataset Exploration

1.1 Dataset Features and Possible Values

The dataset consists of 9000 entries with 11 columns, including 8 numerical features, 2 categorical features, and 1 target variable.

Numerical Features:

- The numerical columns contain continuous values with varying distributions.
- feature_3 and feature_6, have missing values (400 and 500 missing entries, respectively)
- The values range from extreme negatives to positives, indicating possible outliers.
- The mean values are close to zero, suggesting a normalized dataset.

Categorical Features:

category_1 has four unique values:

- Above Average – 1,727 entries
- Below Average – 1,708 entries
- High – 2,763 entries
- Low – 2,802 entries

category_2 represents three regions (A, B, and C), with Region B having the highest frequency.

Target Variable:

- The target variable is binary (0 or 1), with nearly equal distribution (4721 entries for 0 and 4279 for 1), indicating a relatively balanced dataset.

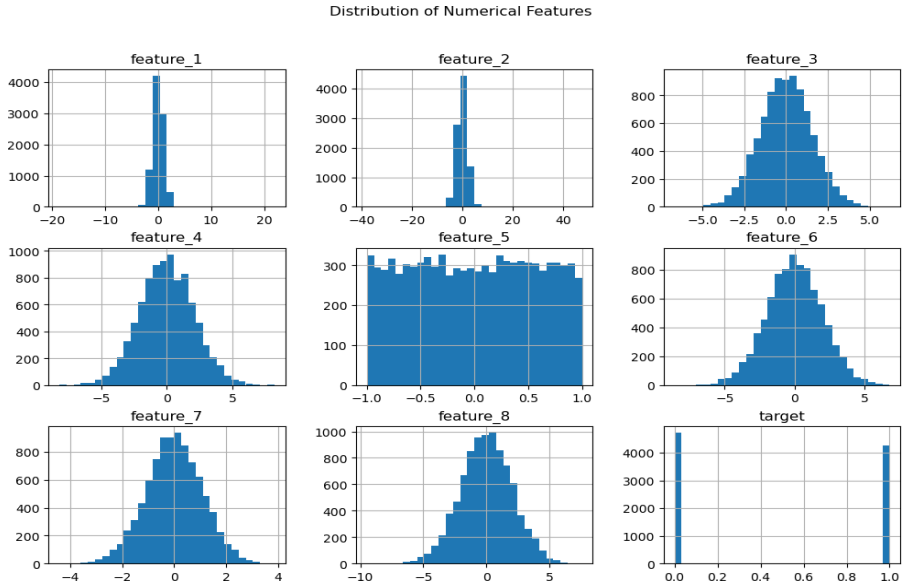
1.2 Dataset Distribution

Visualization through histograms reveal key insights into the numerical dataset's features:

Histogram

- Feature_3, feature_4, feature_6, feature_7, and feature_8 resemble a symmetric distribution.

- Feature_5 shows a uniform distribution.
- Feature_1 and feature_2 are concentrated around zero with potential outliers.
- The target variable is binary (0 and 1), confirming a classification problem.



1.3 Relationship between features

To analyze feature dependencies, a correlation matrix was generated.

1. **Feature-to-Feature relationships:**

Strong positive correlations:

- feature_1 and feature_2 show a perfect correlation (1.00).
- feature_6 is highly correlated with feature_7 and feature_.
- feature_7 and feature_8 also exhibit strong correlation (0.89).

Moderate correlations:

- feature_1 and feature_4 have a moderate positive correlation (0.83).
- feature_2 and feature_4 exhibit a similar moderate correlation.
- feature_5 and feature_7 show a mild correlation (0.25).

Notable negative correlations:

- feature_3 shows strong negative correlations with Feature_1 (-0.83) and Feature_2 (-0.82).
- feature_3 is also highly negatively correlated with Feature_4.

2. **Relationships with Target variable:**

Strong Positive Predictors:

- feature_4 is the strongest positive predictor (0.69).

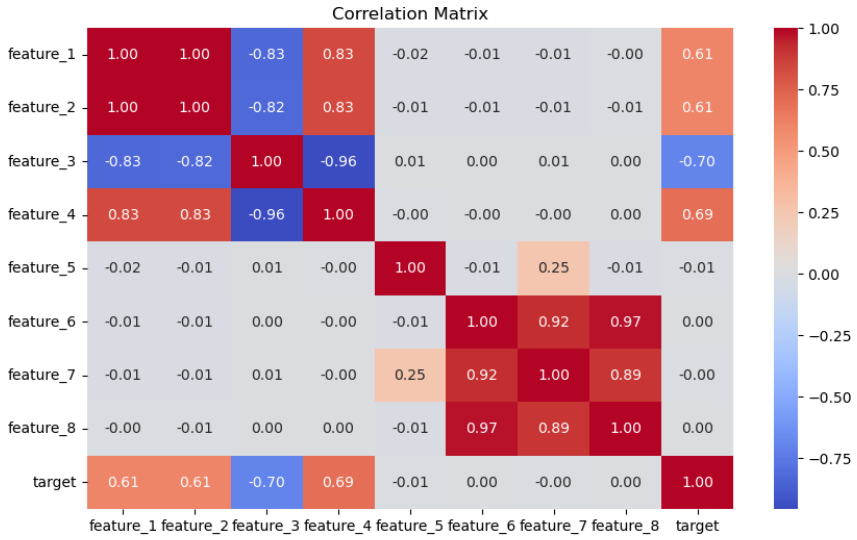
- feature_1 and Feature_2 show a substantial positive correlation with the target (0.61 each).

Strong Negative Predictor:

- Feature_3 is the strongest negative predictor (-0.70).

Weak/Negligible Correlations:

- Feature_5 through Feature_8 show very weak correlations with the target (all near 0).



This correlation analysis highlights feature_4 as the strongest positive predictor, while feature_3 has the most significant negative impact. feature_1 and feature_2 also show a strong positive relationship with the target. Meanwhile, feature_6, feature_7, and feature_8 are highly correlated, suggesting possible redundancy. In contrast, feature_5 through feature_8 have minimal influence on the target. These insights can guide feature selection to enhance model performance.

2. Data preprocessing

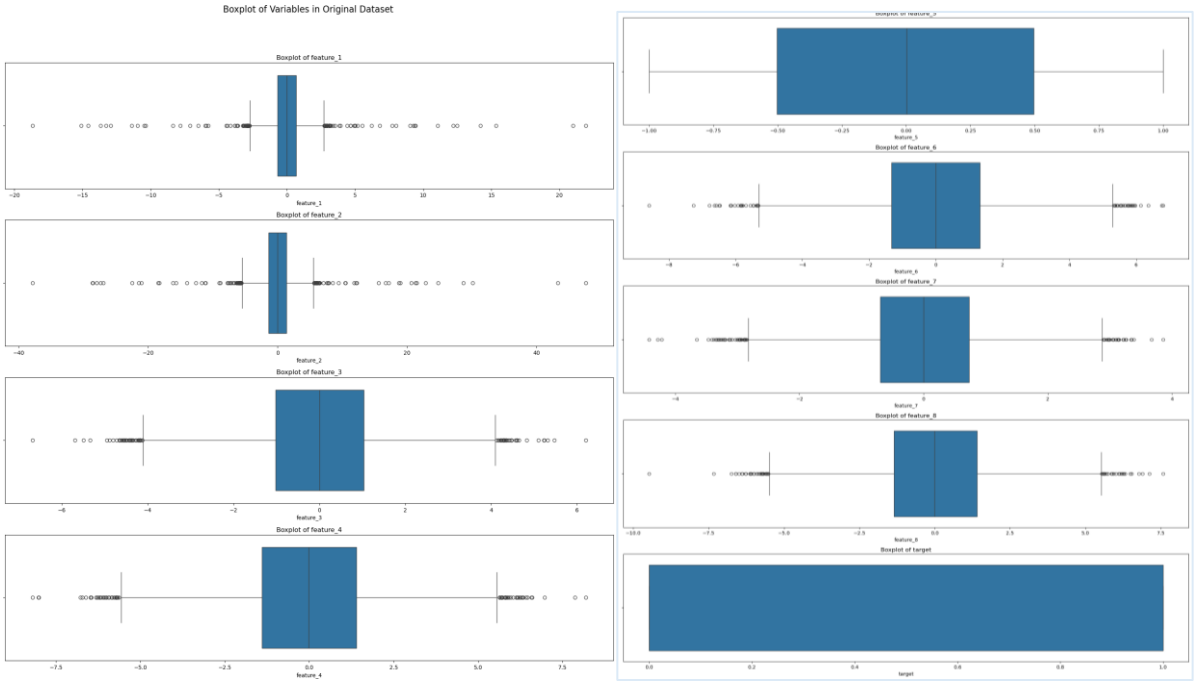
2.1 Handle missing values

The dataset initially shows missing values in feature_3 (400 missing) and feature_6 (500 missing), while all other features and the target variable have no missing data. To handle these missing values, the **SimpleImputer** with the "mean" strategy was applied to the numerical features, filling the missing values with the mean of each respective feature. After the imputation, the dataset shows no remaining missing values across all features and the target variable.

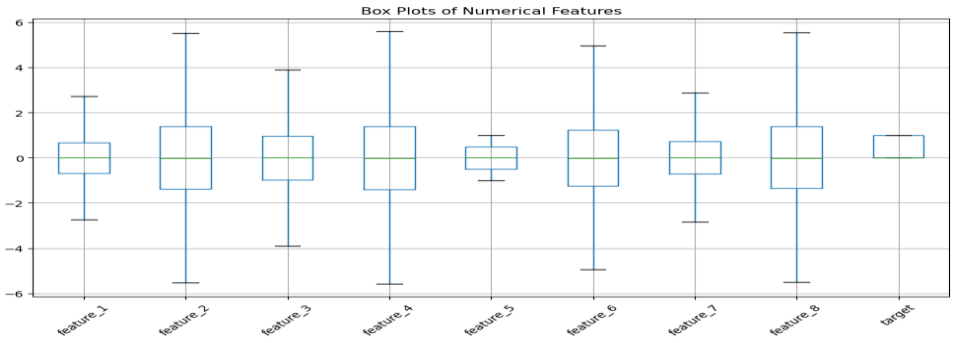
2.2 Detect and treat outliers

The boxplot analysis initially revealed significant outliers in several numerical features, including feature_1, feature_2, feature_6, feature_4, feature_7, and feature_8. Some features

showed a uniform or skewed distribution, while the target variable was binary, indicating a classification problem.



- To detect outliers, the IQR method was applied, and the following features had outliers: feature_3, feature_4, feature_6, feature_7, feature_8: Various indices with outliers were identified, with feature_6 containing 119 outliers.
- The outliers were then treated using the clip_outliers function, which applied the IQR method to clip values outside the lower and upper bounds. After clipping the outliers, a new boxplot was generated, showing that all features were now free of outliers.



2.3 Encode categorical variables

The categorical variables category_1 and category_2 were encoded using Ordinal Encoding and One-Hot Encoding, respectively, to make them suitable for machine learning models.

For category_1
The variable category_1 had an inherent order: ['Low', 'Below Average', 'Above Average', 'High']. Since this variable represents an ordinal categorical feature, we use **Ordinal Encoding** to convert it into numerical values that reflect its ranking.

- 'Low' → 0
- 'Below Average' → 1
- 'Above Average' → 2
- 'High' → 3

The newly created category_1_encoded column replaced category_1 in the dataset.

For category_2
One-Hot Encoding is applied to category_2 because it is a nominal variable with no inherent ranking, and one-hot encoding prevents the model from misinterpreting relationships between categories.

- category_2_Region B
- category_2_Region C

The original category_2 column was dropped after encoding.

2.4 Scaling

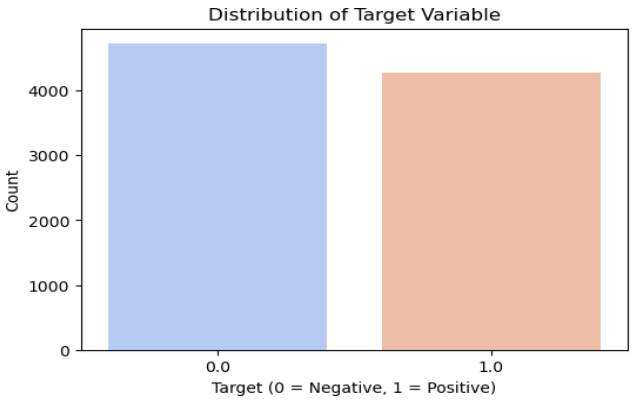
To maintain consistency across features and enhance model performance, numerical features were scaled using **StandardScaler.** This transformation adjusts the features to have a mean of 0 and a standard deviation of 1, ensuring they are on a uniform scale.

The final dataset is now fully cleaned, with no outliers or missing values, all categorical features encoded, and numerical features standardized.

3. Explanatory Data Analysis (EDA)

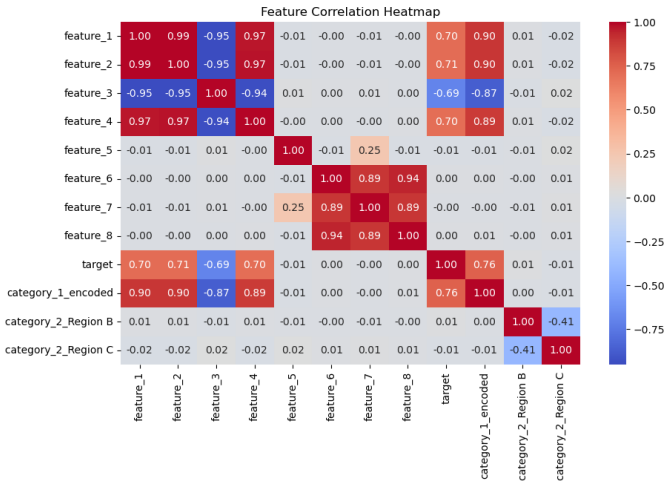
3.1 Key insights from data exploration

Distribution of target variable
The count plot was used to analyze the distribution of the target variable, providing insights into class balance. The target variable has two classes: 0 (Negative) and 1 (Positive), with the 'Negative' class being more frequent. The imbalance could lead to machine learning models being biased toward the majority class, making accurate predictions for the minority class more challenging.



Feature correlation heatmap

A correlation heatmap was created to analyze the relationships between numerical features.

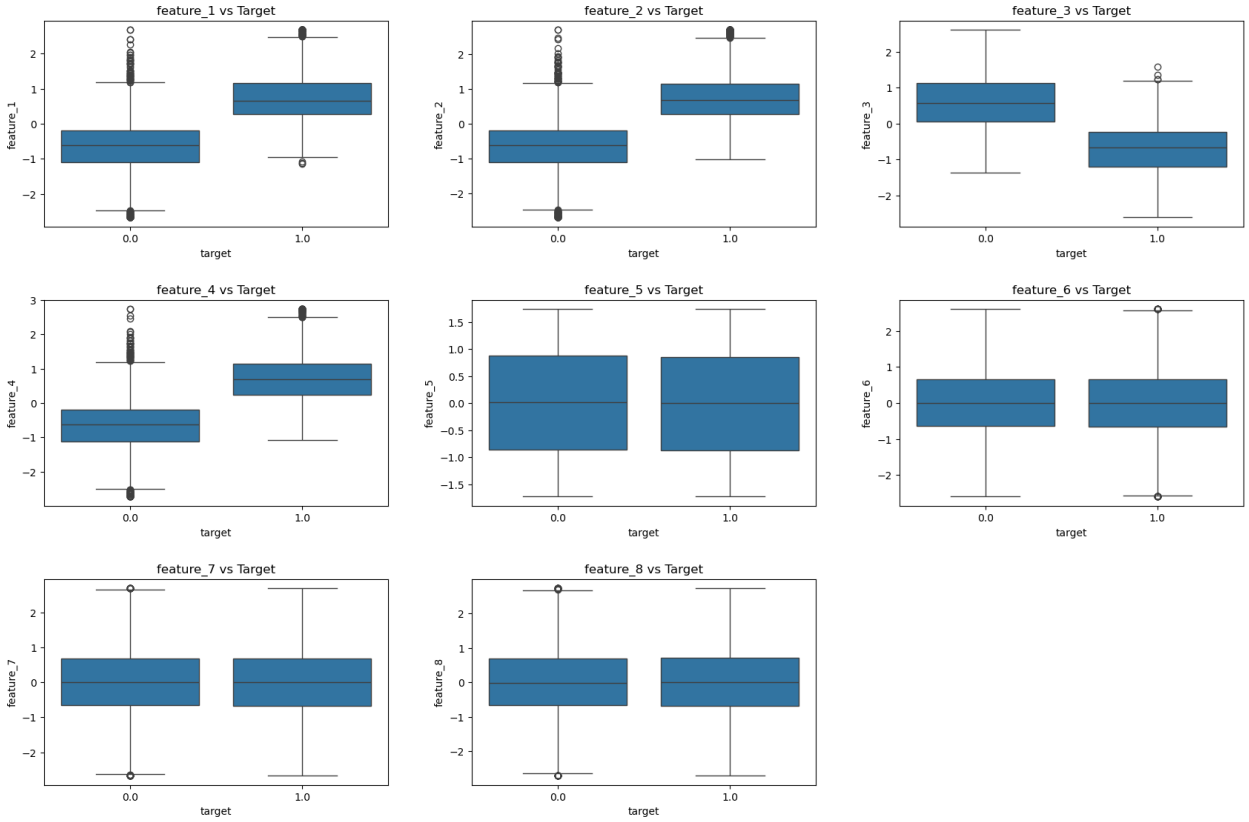


The heatmap shows that features 1, 2, and 4 have strong correlations with the target variable, indicating they are significant predictors of the target value. In contrast, feature_3, features 5-8 and categorical feature 2 (regions B and C) exhibit weak correlations with the target, suggesting limited predictive power. However, categorical feature 1 stands out as an important predictor of the target value.

3.2 Feature-target relationships

Boxplots for numerical features vs. target variable

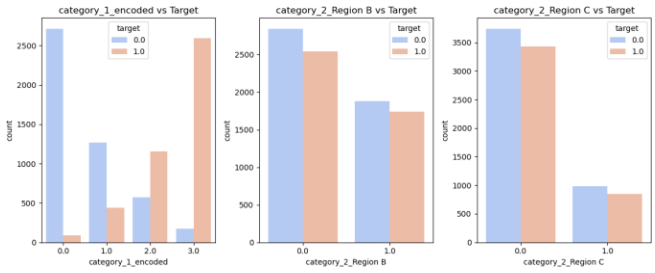
Box plots were generated for each numerical feature to compare their distributions across target groups.



Each box plot displays the distribution of a numerical feature across the two categories of the target variable. feature_1, feature_2, feature_3, and feature_4 show the strongest predictive potential due to clear differences in distribution between target groups.

Countplots for categorical features vs Target Variable

The relationship between categorical features and the target variable was analyzed using count plots. The analysis revealed that Category_1_encoded and 'Region B' in Category_2 may serve as strong predictors, with higher values in Category_1_encoded and the presence of 'Region B' associated with a higher chance of a positive outcome. In contrast, 'Region C' appears to have a weaker impact.

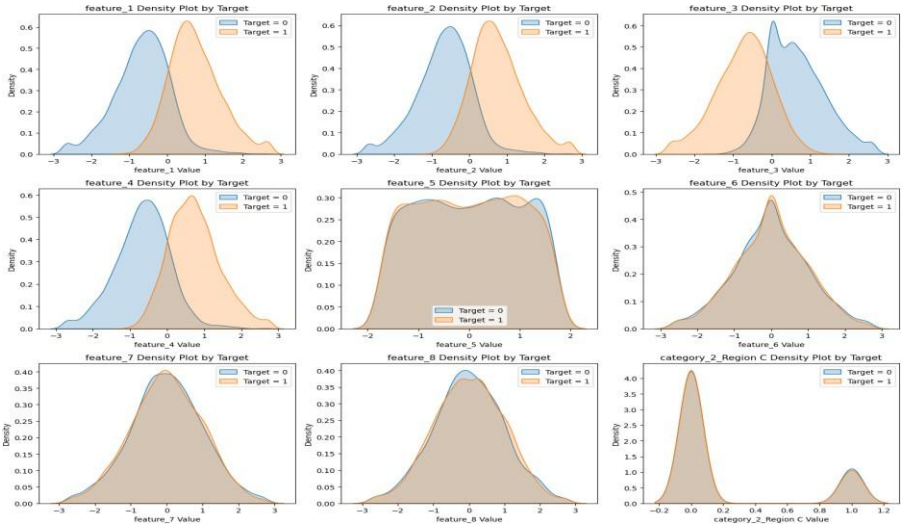


3.3 Hypothesis Testing

Here the objective is to determine if the features are significant predictors for target classes.

Methods:

- T-Test for numerical features: The independent samples t-test was used to assess whether there was a significant difference in the means of the numerical features between the two target variables (0 and 1).
- Chi-Square test for categorical features: The chi-square test was used to evaluate the association between categorical features (Category 1 and Category 2) and the target variable.

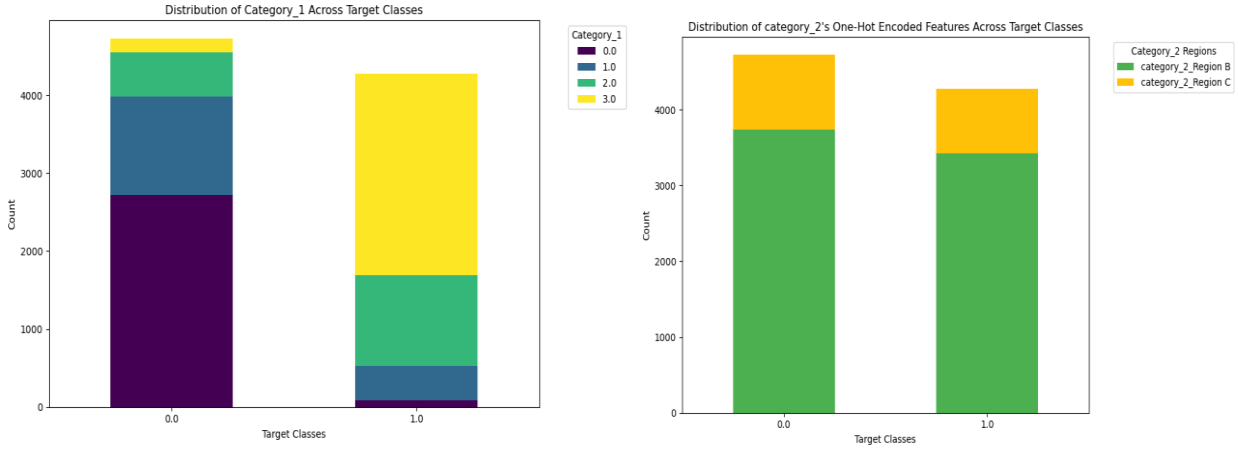


Numerical Features:

- Feature 1, 2, 3, and 4: The p-values were significantly less than 0.05 for these features, indicating a significant difference in their means between the two target variables. These features are useful for prediction.
- Feature 5, 6, 7, and 8: The p-values were greater than 0.05 for these features, suggesting no significant difference between the target variables. These features are likely not useful for prediction.

Categorical Features:

- Category 1: The Chi-Square test resulted in a p-value significantly less than 0.05, indicating that there is a strong association between Category 1 and the target variables.
- Category 2: The Chi-Square test resulted in a p-value greater than 0.05, indicating no significant association between Category 2 and the target variables.



4. Feature engineering

Feature engineering was performed on the dataset to enhance predictive performance by creating new features and selecting the most relevant ones. Various transformations, interactions, and statistical tests were applied to extract meaningful information.

4.1 Feature engineering process

- Feature Interactions: Pairwise multiplication, addition, subtraction, and division were applied among selected features (feature_5, feature_6, feature_7, feature_8).
- Polynomial Features: Square and cube transformations were created for selected features.
- Binning: Continuous variables were discretized into quartile-based bins.

Feature selection

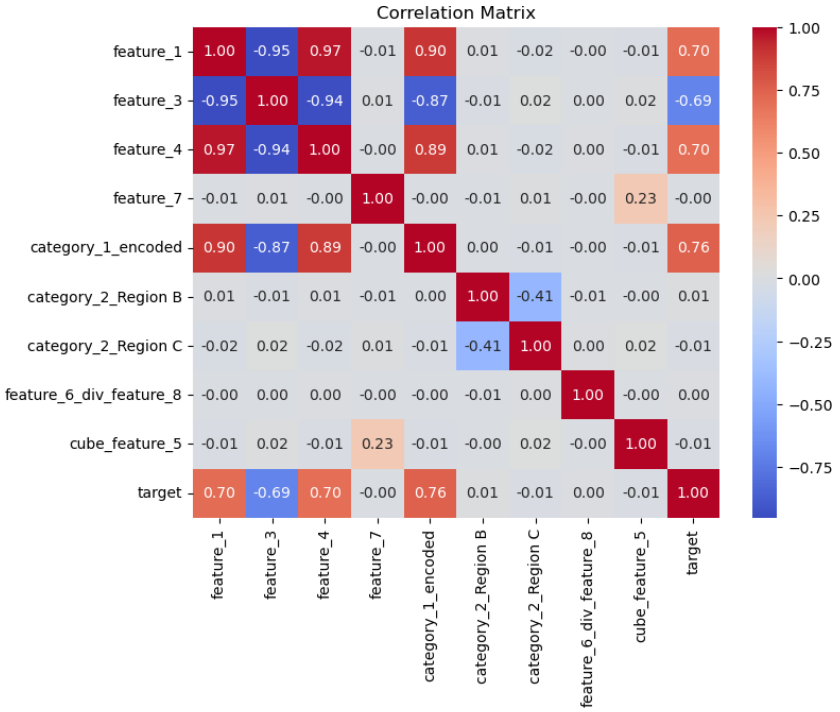
- A t-test was performed to evaluate the statistical significance of each feature in distinguishing between target classes.
- Features with the lowest p-values were considered most relevant.

The final selected features included:

- feature_1, feature_3, feature_4, feature_7
- category_1_encoded, category_2_Region B, category_2_Region C
- feature_6_div_feature_8, cube_feature_5
- target

	feature_1	feature_3	feature_4	feature_7	category_1_encoded	category_2_Region B	category_2_Region C	feature_6_div_feature_8	cube_feature_5	target
0	0.488558	-0.437472	0.410834	-1.218681	2.0	0.0	1.0	1.046159	2.533330	1.0
1	-0.134837	-0.000293	0.201208	-1.255315	1.0	0.0	0.0	1.365752	1.834811	0.0
2	0.636778	-0.515016	0.838285	1.113257	3.0	0.0	1.0	1.394568	-0.079102	1.0
3	1.496152	-1.653667	1.297736	1.984733	3.0	1.0	0.0	1.080484	3.593533	1.0
4	-0.228977	0.383965	-0.211671	0.874965	1.0	0.0	1.0	-0.569593	1.195410	0.0

4.2 Correlation analysis



A heatmap was generated to visualize correlations between the selected features and the target variable:

- category_1_encoded showed the highest correlation with the target (0.76).
- feature_1, and feature_4 also exhibited strong correlations.
- Weakly correlated or redundant features were excluded.

The feature engineering process resulted in a refined dataset with improved feature representation. The selected features will be used for model training to enhance prediction accuracy.

5. Modeling

5.1 Data preparation

The dataset was divided into features (X) and the target variable (y). The data was then split into training and testing sets using an 80-20 split to ensure a robust evaluation. The stratification technique was used to maintain class distribution across training and testing sets.

5.2 Models trained

The following machine learning models were trained and evaluated:

- Logistic Regression
- Support Vector Machine (SVM)
- Random Forest Classifier
- Decision Tree Classifier
- Gradient Boosting Classifier
- Bagging Classifier
- XGBoost Classifier

5.3 Model evaluation metrics

Each model was evaluated using key performance metrics, including:

- Accuracy: Overall correctness of the model.
- Precision: Proportion of true positives among all predicted positives.
- Recall: Proportion of true positives identified out of all actual positives.
- F1-Score: Harmonic mean of precision and recall.
- ROC-AUC: Area under the Receiver Operating Characteristic curve, assessing the model's ability to distinguish between classes.
- Cross-validation with 10 folds was performed to ensure robustness, and the mean cross-validation error (Root Mean Squared Error - RMSE) was calculated.

5.4 Results

1. Logistic Regression

- Accuracy: 85.89%
- Precision: 86.07%
- Recall: 85.89%
- F1 Score: 85.90%
- ROC-AUC: 93.24%

Cross-Validation:

- Mean RMSE: 0.3791

Logistic Regression performed well, but its linear nature may limit its ability to capture complex patterns in the data.

2. Support Vector Machine (SVM)

- Accuracy: 86.22%
- Precision: 86.39%
- Recall: 86.22%
- F1 Score: 86.23%
- ROC-AUC: 93.08%

Cross-Validation:

- Mean RMSE: 0.3772

SVM showed slightly better performance than Logistic Regression and was effective in handling non-linear decision boundaries.

3. Random Forest Classifier

- Accuracy: 88.67%
- Precision: 88.69%
- Recall: 88.67%
- F1 Score: 88.65%
- ROC-AUC: 95.71%

Cross-Validation:

- Mean RMSE: 0.3511

Random Forest emerged as one of the best-performing models, leveraging ensemble learning to improve generalization.

4. Decision Tree Classifier

- Accuracy: 84.28%
- Precision: 84.27%
- Recall: 84.28%
- F1 Score: 84.27%
- ROC-AUC: 84.21%

Cross-Validation:

- Mean RMSE: 0.4097

Decision Tree showed lower performance, likely due to overfitting. Regularization techniques such as pruning could improve results.

5. Gradient Boosting Classifier

- Accuracy: 88.22%
- Precision: 88.23%
- Recall: 88.22%
- F1 Score: 88.21%
- ROC-AUC: 95.90%

Cross-Validation:

- Mean RMSE: 0.3431

Gradient Boosting performed competitively with Random Forest, showing strong predictive capability by iteratively correcting errors.

6. Bagging Classifier

- Accuracy: 87.50%
- Precision: 87.56%
- Recall: 87.50%
- F1 Score: 87.48%
- ROC-AUC: 94.45%

Cross-Validation:

- Mean RMSE: 0.3589

Bagging improved model stability and reduced variance, performing slightly below Random Forest and Gradient Boosting.

7. XGBoost Classifier

- Accuracy: 87.67%
- Precision: 87.67%
- Recall: 87.67%
- F1 Score: 87.67%
- ROC-AUC: 95.43%

Cross-Validation:

- Mean RMSE: 0.3608

XGBoost showed strong results, combining boosting techniques for enhanced accuracy and generalization.

5.5 Conclusion

Based on the evaluation results, the Random Forest Classifier achieved the best performance with the highest accuracy (88.67%) and ROC-AUC score (95.71%). Gradient Boosting followed closely, demonstrating competitive accuracy (88.22%) and the highest ROC-AUC score (95.90%).

Key Observations:

- Ensemble methods (Random Forest, Gradient Boosting, XGBoost, and Bagging) outperformed individual models, highlighting their ability to improve predictive performance.
- Logistic Regression and SVM performed well but were outperformed by tree-based models due to their ability to capture non-linear relationships.
- Decision Tree had the lowest performance, likely due to overfitting, which could be mitigated with hyperparameter tuning.

6. Hyperparameter tuning

The goal is to optimize model parameters using GridSearchCV and compare model performance before and after tuning.

6.1 Hyperparameter Grid setup

The following models and their respective hyperparameter grids were considered for tuning:

Logistic Regression

Hyperparameters:

- Regularization Strength (C): [0.1, 1, 10]
- Maximum Iterations: [100, 200, 300]
- Solver: ['liblinear', 'lbfgs']

Support Vector Machine (SVM)

Hyperparameters:

- Regularization Strength (C): [0.1, 1, 10]
- Kernel: ['linear', 'rbf']
- Gamma: ['scale', 'auto']

Random Forest

Hyperparameters**:**

- Number of Trees (n_estimators): [50, 100, 200]
- Maximum Depth: [None, 10, 20]
- Minimum Samples Split: [2, 5, 10]

Decision Tree

Hyperparameters:

- Criterion: ['gini', 'entropy']
- Maximum Depth: [None, 10, 20]
- Minimum Samples Split: [2, 5, 10]

Gradient Boosting

Hyperparameters:

- Number of Trees (n_estimators): [50, 100, 200]
- Learning Rate: [0.01, 0.1, 0.2]
- Maximum Depth: [3, 5, 10]

Bagging Classifier
Hyperparameters:

- Number of Base Learners (n_estimators): [10, 50, 100]
- Maximum Samples: [0.5, 0.7, 1.0]

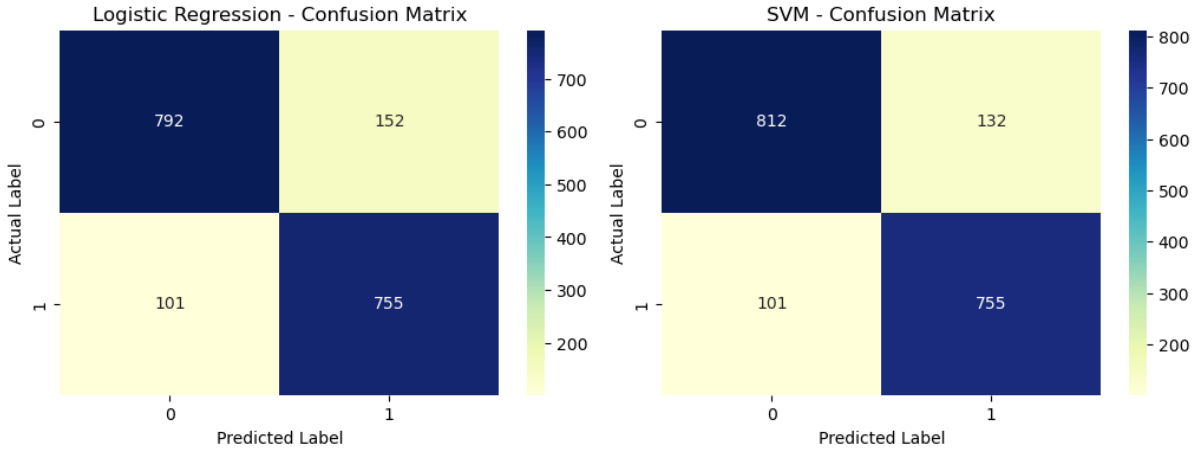
XGBoost
Hyperparameters:

- Number of Trees (n_estimators): [50, 100, 200]
- Learning Rate: [0.01, 0.1, 0.2]
- Maximum Depth: [3, 5, 10]

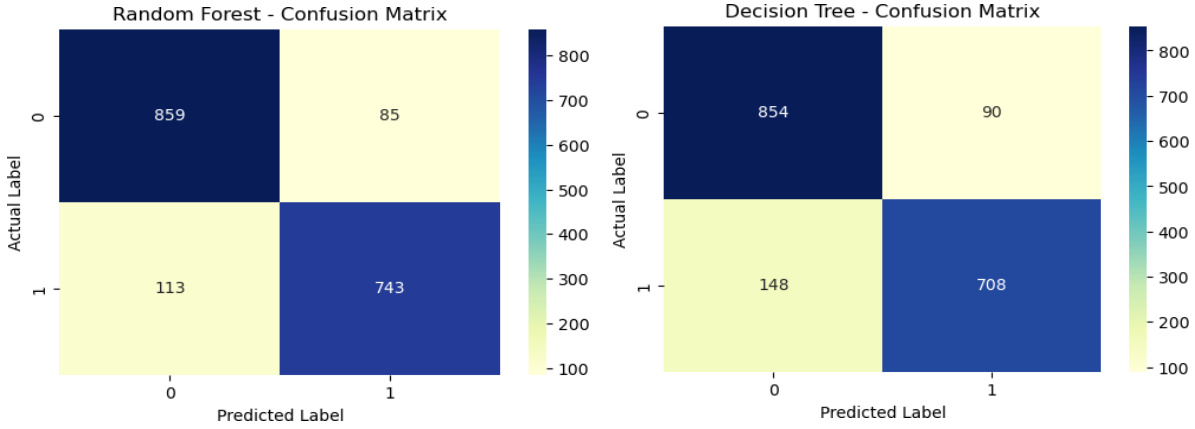
Model Training and Evaluation
Each model underwent hyperparameter tuning using GridSearchCV (3-fold cross-validation). Post-tuning, the best hyperparameters were applied to the test dataset for evaluation. The key performance metrics recorded include:

- Accuracy
- Precision
- Recall
- F1 Score
- ROC-AUC Score

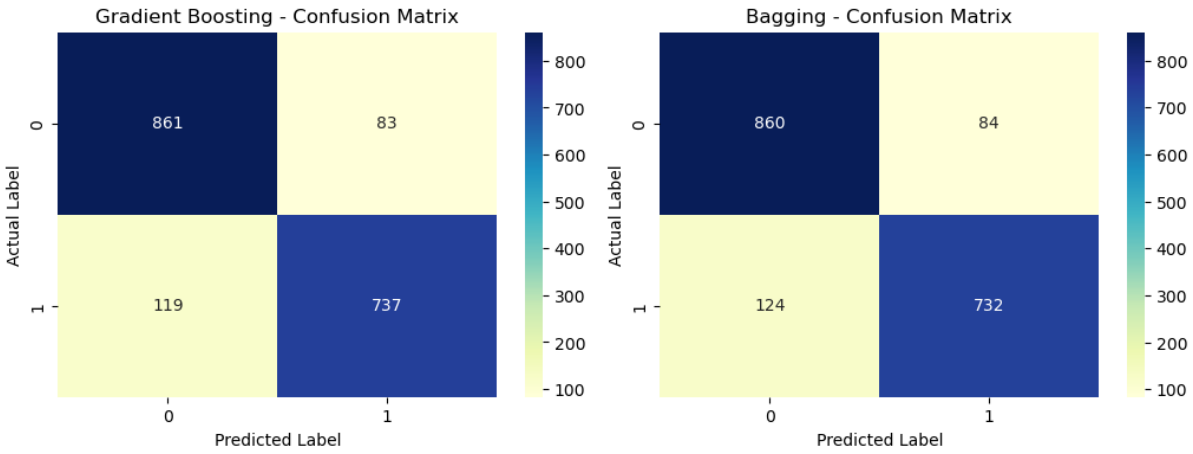
Model	Accuracy (Before)	Accuracy (After)
Random Forest	88.67%	89%
XGBoost	87.67%	89%
Gradient Boosting	88.22%	89%
Bagging	87.50%	88%
SVM	86.22%	87%
Decision Tree	84.28%	87%
Logistic Regression	85.89%	86%

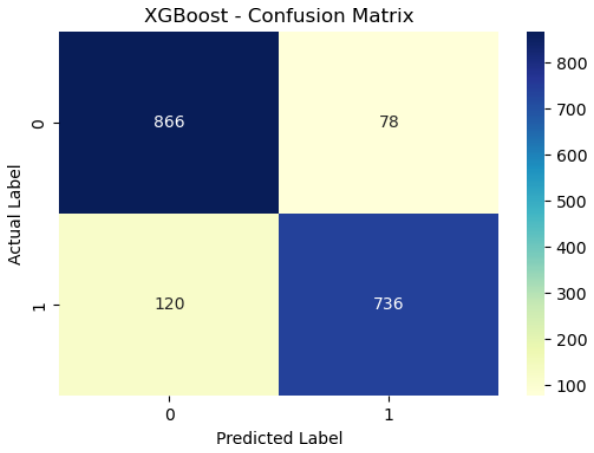

Logistic Regression - Confusion Matrix
792
152
101
755
Actual Label
Predicted Label
0
1
700
600
500
400
300
200

SVM - Confusion Matrix
812
132
101
755
Actual Label
Predicted Label
0
1
800
700
600
500
400
300
200


Random Forest - Confusion Matrix
859
85
113
743
Actual Label
Predicted Label
0
1
800
700
600
500
400
300
200
100

Decision Tree - Confusion Matrix
854
90
148
708
Actual Label
Predicted Label
0
1
800
700
600
500
400
300
200
100





From the tuning results, it is evident that Random Forest, XGBoost, and Gradient Boosting are the top-performing models, all achieving an accuracy of 89%. These models were able to strike a good balance between accuracy, precision, recall, F1 score, and ROC-AUC. The Logistic Regression model performed the lowest in comparison, with an accuracy of 86%, but still showed reasonable classification results.

These results suggest that ensemble methods like Random Forest, XGBoost, and Gradient Boosting offer robust performance, and would be good candidates for production models in this dataset.

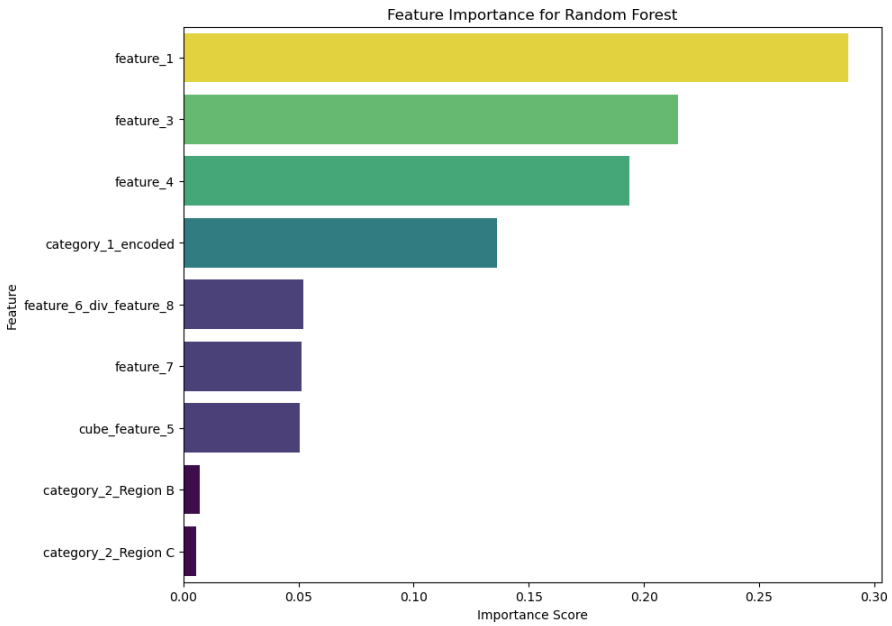
7. Model interpretation

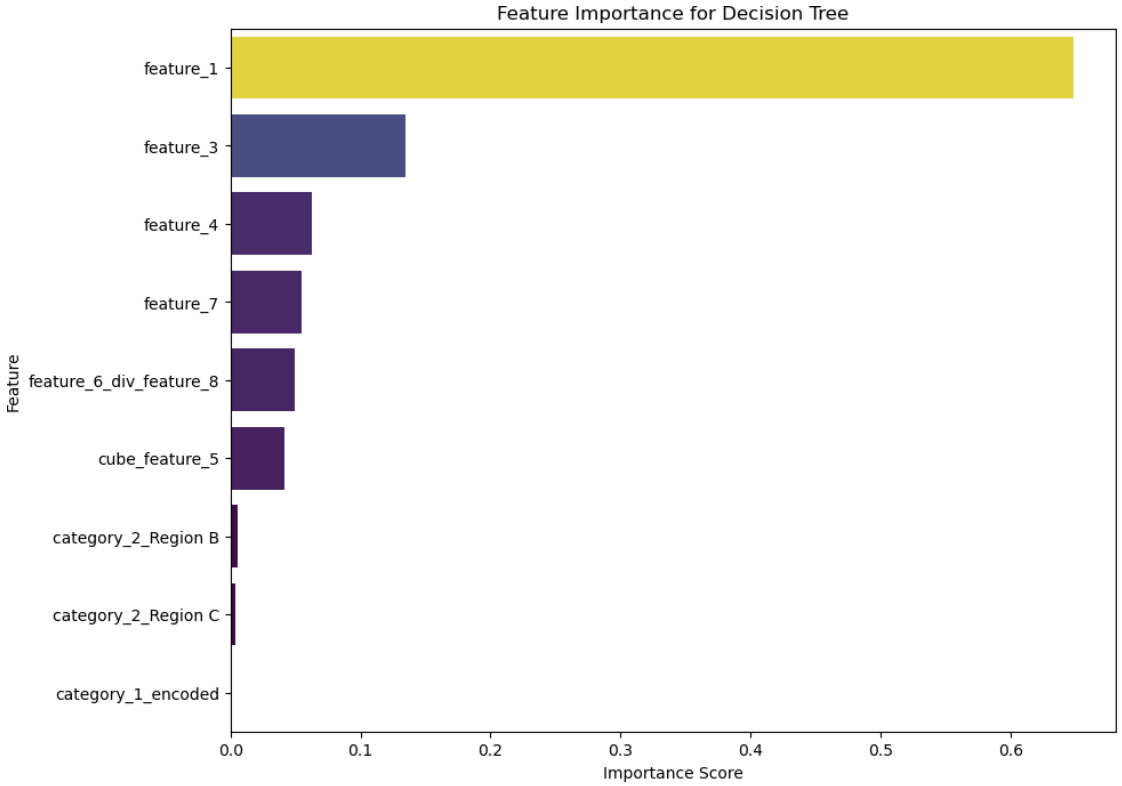
7.1 Feature Importance Analysis

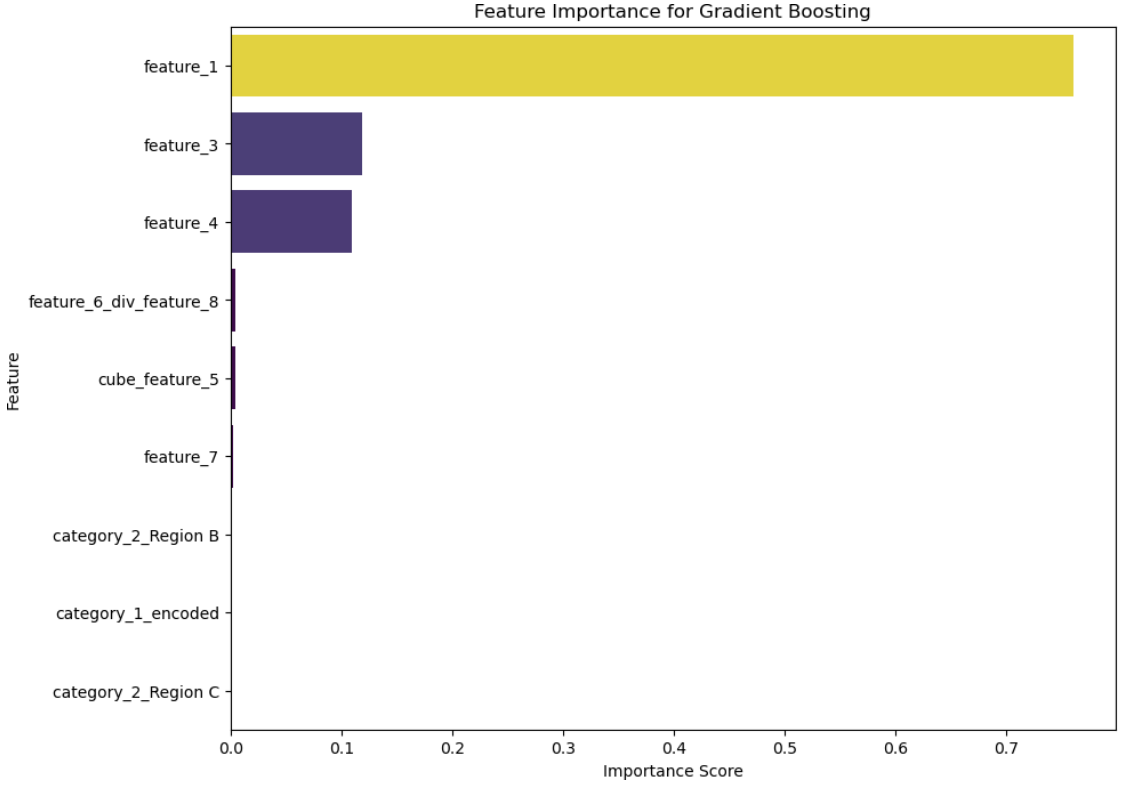
Feature importance provides insights into which features contribute most to a model's predictions. The function plot_feature_importance() is used to visualize the importance scores of features in models that support feature importance, such as tree-based models.

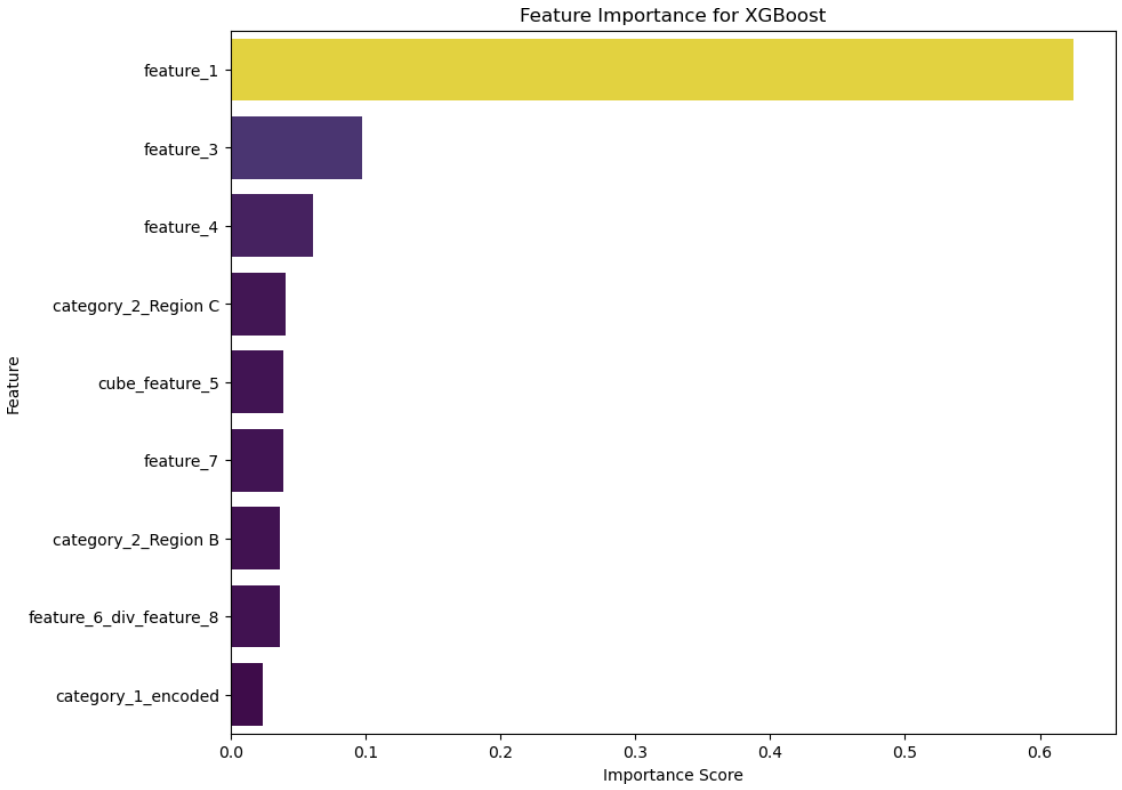
Findings

- Logistic Regression & SVM: These models do not have a feature importance attribute and were skipped in feature importance analysis.
- Random Forest: Feature importance analysis identified feature_1 as the most significant, followed by feature_3and feature_4.
- Gradient Boosting: The model highlighted feature_1 as having the highest importance, followed by other influential features.
- XGBoost: Similarly, feature_1 was found to be the most crucial, with feature_3 and feature_4 also contributing significantly.
- Bagging Model: This model does not provide feature importance directly and was therefore skipped.




Feature Importance for Decision Tree
feature_1
feature_3
feature_4
feature_7
feature_6_div_feature_8
cube_feature_5
category_2_Region B
category_2_Region C
category_1_encoded
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Importance Score


Feature Importance for Gradient Boosting
feature_1
feature_3
feature_4
feature_6_div_feature_8
cube_feature_5
feature_7
category_2_Region B
category_1_encoded
category_2_Region C
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Importance Score

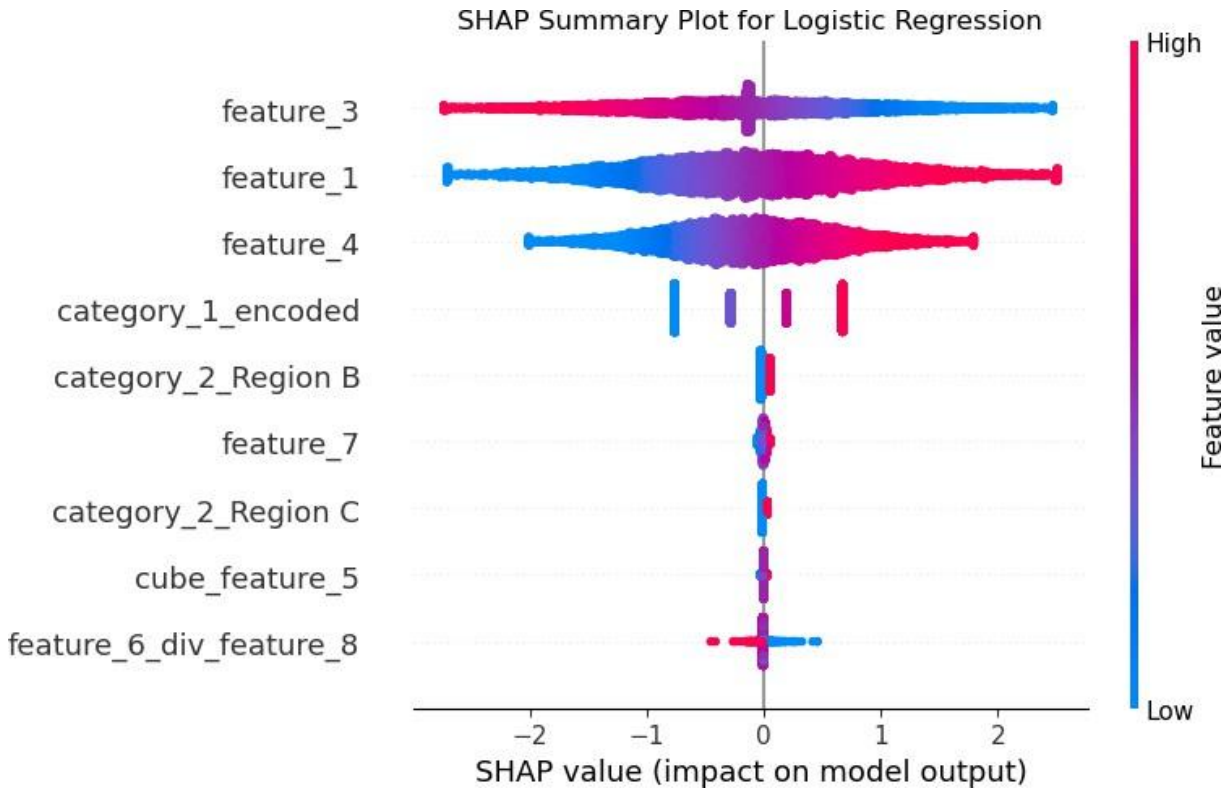


7.2 SHAP analysis

SHAP values were then analyzed to further interpret feature importance from the models. High-importance features across different models were identified, offering a more detailed breakdown of their contributions.

Findings

- Tree-based models (Random Forest, Decision Tree, Gradient Boosting, XGBoost): SHAP values confirmed that feature_1 had the most significant impact, followed by feature_3 and feature_4.
- Logistic Regression: SHAP's Linear Explainer was used for interpretation.
- Kernel Explainer: Applied to models like SVM and Bagging for feature impact estimation.



Conclusion

The analysis confirms that tree-based models offer direct feature importance measures, while SHAP values provide a consistent interpretation across different models. The most influential features across models were feature_1, feature_3, and feature_4, demonstrating their predictive value.

Future Work

- Investigate interaction effects between features.
- Utilize permutation importance for additional validation.
- Apply SHAP explanations to real-world datasets for business insights.

This report serves as a guide to understanding model decisions, enhancing transparency and trust in machine learning applications.