# **Project Step 4: Model Building and Evaluation**

“**Problem topic selected:** Our team will build a classifier that classifies whether a given school is high- or low-performing based on its ELA (English Language) or Math score.”

1.

Link to the dataset:

<https://docs.google.com/spreadsheets/d/1VNFo5skR5iku3vJPTmZRAr4BJBN5NAmWhiHDckggb48/edit#gid=0>

Dataset was saved as the CSV file (Data Collection-Project.csv)

**2.**

**Model Building and Evaluation:**

**Code:**

**#ASU ID: 1231052367**

**#Sumasree Battula**

**#Group number: Sparky Big data Team**

**#Model Building and evaluation**

**#Using the Gradient boosting classifier**

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.preprocessing import LabelEncoder

# Load data

dfcp = pd.read\_csv('C:/Users/sumas/Downloads/Data Collection-Project.csv')

# Data cleaning and preprocessing

dfcp = dfcp.drop(["Free lunch Not Eligible"], axis=1)

dfcp = dfcp.dropna()

# Removing duplicates

dfcp.drop\_duplicates(subset=['School Name', 'Zip Code'], keep='first', inplace=True)

# Replace non-numeric characters in specified columns

dfcp['Math Score'] = dfcp['Math Score'].str.replace('%', '').astype(float) / 100

dfcp['English Score'] = dfcp['English Score'].str.replace('%', '').astype(float) / 100

dfcp['Graduation Rate'] = dfcp['Graduation Rate'].str.replace('%', '').astype(float) / 100

# Define high-performing schools based on median scores for English, Math, and Graduation Rate

medians = dfcp[['English Score', 'Math Score', 'Graduation Rate']].median()

dfcp['High\_Performing'] = ((dfcp['English Score'] > medians['English Score']) |

(dfcp['Math Score'] > medians['Math Score']) |

(dfcp['Graduation Rate'] > medians['Graduation Rate'])).astype(int)

# Encode categorical data

categorical\_columns = dfcp.select\_dtypes(include=['object']).columns.tolist()

for column in categorical\_columns:

dfcp[column] = LabelEncoder().fit\_transform(dfcp[column])

# Prepare features and target

X = dfcp.drop(['High\_Performing'], axis=1)

y = dfcp['High\_Performing']

# Initialize a Gradient Boosting Classifier

classifier = GradientBoostingClassifier(random\_state=42)

# Perform K-Fold Cross Validation

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_accuracy = cross\_val\_score(classifier, X, y, cv=kf, scoring='accuracy')

cv\_precision = cross\_val\_score(classifier, X, y, cv=kf, scoring='precision')

cv\_recall = cross\_val\_score(classifier, X, y, cv=kf, scoring='recall')

cv\_f1 = cross\_val\_score(classifier, X, y, cv=kf, scoring='f1')

# Print cross-validated metrics

print("Cross-validated Metrics:")

print(f"Accuracy: {cv\_accuracy.mean():.2f}")

print(f"Precision: {cv\_precision.mean():.2f}")

print(f"Recall: {cv\_recall.mean():.2f}")

print(f"F1-score: {cv\_f1.mean():.2f}")

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the model on the training set

classifier.fit(X\_train, y\_train)

# Predict on the test set

predictions = classifier.predict(X\_test)

# Evaluate the model on the test set

accuracy = accuracy\_score(y\_test, predictions)

report = classification\_report(y\_test, predictions)

# Print model accuracy and classification report

print("\nEvaluation on Test Set:")

print(f"Model Accuracy: {accuracy \* 100:.2f}%")

print("Classification Report:")

print(report)

# Predict on the entire dataset

dfcp['Predicted\_Performance'] = classifier.predict(X)

# Map predicted labels back to 'High-Performing' or 'Low-Performing'

dfcp['Performance\_Label'] = dfcp['Predicted\_Performance'].map({1: 'High-Performing', 0: 'Low-Performing'})

# Output the DataFrame with the new predictions and their labels

print("\nPredictions on Entire Dataset:")

print(dfcp[['School Name', 'Math Score', 'English Score', 'Graduation Rate', 'Performance\_Label']])

# Save the results to an Excel file

excel\_path = 'C:/Users/sumas/Downloads/Project\_IFT511/IFTSchool\_Performance\_Predictions\_Gradient\_boosting.xlsx'

dfcp.to\_excel(excel\_path, index=False)

Code snippet:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

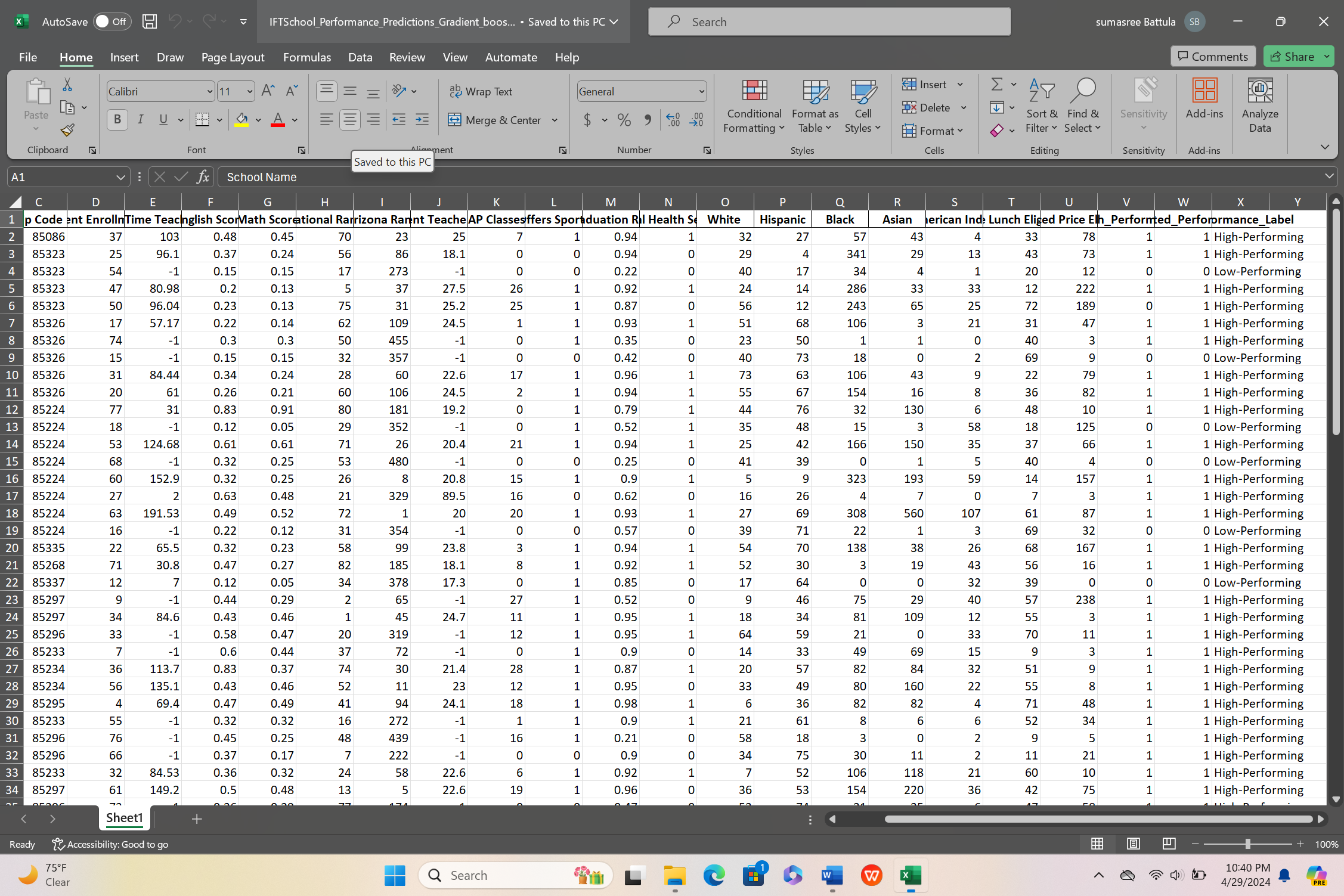
A screenshot of a computer

Description automatically generated

Output snippet and Excel created snippet:

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

Explanation of the model:

The above Python script illustrates how machine learning can be utilized to categorize schools based on their performance indicators. It initiates with data loading and preprocessing steps, ensuring data quality by eliminating duplicates and converting non-numeric data into usable formats. Through feature engineering and label encoding, categorical data is transformed to facilitate model compatibility.

Overview of Gradient Boosting Classifier Model:

Employing a Gradient Boosting Classifier, the model demonstrates promising performance metrics during cross-validation, with an average accuracy of 89%. Precision, recall, and F1-score metrics further indicate the model's strong predictive ability, particularly in identifying high-performing schools. However, evaluation on the test set reveals a reduction in accuracy to 76.47%, suggesting potential overfitting or limited generalization to unseen data.

Key Findings:

* Cross-validated Metrics: The model exhibits robust performance metrics, including precision (90%), recall (97%), and F1-score (93%), indicating reliable predictions during cross-validation.
* Evaluation on Test Set: While precision remains high for both high and low-performing schools, recall for low-performing schools is notably low, suggesting potential limitations in certain school category identification by the model.
* Predictions on Entire Dataset: The model's predictions on the entire dataset showcase the classification of schools as high or low-performing based on learned patterns. However, validation against real-world data may be necessary to confirm the accuracy of these classifications.

Conclusion:

Although the model shows proficiency in predicting school performance, disparities between cross-validation and test set outcomes suggest potential areas for enhancement, such as mitigating overfitting or improving model generalization. Further validation and refinement of the model may be required to ensure its effectiveness in practical scenarios.

**Extra credit:**

**Model rebuilding: Random forest classification**

**Code:**

#ASU ID : 1231052367

#Sumasree Battula

#Group number : Sparky Big data Team

#Model Building and evaluation

#Using the random forest classifier

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, KFold

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.preprocessing import LabelEncoder

# Load data

dfcp = pd.read\_csv('C:/Users/sumas/Downloads/Data Collection-Project.csv')

# Data cleaning and preprocessing

dfcp = dfcp.drop(["Free lunch Not Eligible"], axis=1)

dfcp = dfcp.dropna()

# Removing duplicates

dfcp.drop\_duplicates(subset=['School Name', 'Zip Code'], keep='first', inplace=True)

# Replace non-numeric characters in specified columns

dfcp['Math Score'] = dfcp['Math Score'].str.replace('%', '').astype(float) / 100

dfcp['English Score'] = dfcp['English Score'].str.replace('%', '').astype(float) / 100

dfcp['Graduation Rate'] = dfcp['Graduation Rate'].str.replace('%', '').astype(float) / 100

# Define high-performing schools based on median scores for English, Math, and Graduation Rate

medians = dfcp[['English Score', 'Math Score', 'Graduation Rate']].median()

dfcp['High\_Performing'] = ((dfcp['English Score'] > medians['English Score']) |

(dfcp['Math Score'] > medians['Math Score']) |

(dfcp['Graduation Rate'] > medians['Graduation Rate'])).astype(int)

# Encode categorical data

categorical\_columns = dfcp.select\_dtypes(include=['object']).columns.tolist()

for column in categorical\_columns:

dfcp[column] = LabelEncoder().fit\_transform(dfcp[column])

# Prepare features and target

X = dfcp.drop(['High\_Performing'], axis=1)

y = dfcp['High\_Performing']

# Initialize a Random Forest Classifier

classifier = RandomForestClassifier(random\_state=42)

# Perform K-Fold Cross Validation

kf = KFold(n\_splits=5, shuffle=True, random\_state=42)

cv\_accuracy = cross\_val\_score(classifier, X, y, cv=kf, scoring='accuracy')

cv\_precision = cross\_val\_score(classifier, X, y, cv=kf, scoring='precision')

cv\_recall = cross\_val\_score(classifier, X, y, cv=kf, scoring='recall')

cv\_f1 = cross\_val\_score(classifier, X, y, cv=kf, scoring='f1')

# Print cross-validated metrics

print("Cross-validated Metrics:")

print(f"Accuracy: {cv\_accuracy.mean():.2f}")

print(f"Precision: {cv\_precision.mean():.2f}")

print(f"Recall: {cv\_recall.mean():.2f}")

print(f"F1-score: {cv\_f1.mean():.2f}")

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the model on the training set

classifier.fit(X\_train, y\_train)

# Predict on the test set

predictions = classifier.predict(X\_test)

# Evaluate the model on the test set

accuracy = accuracy\_score(y\_test, predictions)

report = classification\_report(y\_test, predictions)

# Print model accuracy and classification report

print("\nEvaluation on Test Set:")

print(f"Model Accuracy: {accuracy \* 100:.2f}%")

print("Classification Report:")

print(report)

# Predict on the entire dataset

dfcp['Predicted\_Performance'] = classifier.predict(X)

# Map predicted labels back to 'High-Performing' or 'Low-Performing'

dfcp['Performance\_Label'] = dfcp['Predicted\_Performance'].map({1: 'High-Performing', 0: 'Low-Performing'})

# Output the DataFrame with the new predictions and their labels

print("\nPredictions on Entire Dataset:")

print(dfcp[['School Name', 'Math Score', 'English Score', 'Graduation Rate', 'Performance\_Label']])

# Save the results to an Excel file

excel\_path = 'C:/Users/sumas/Downloads/Project\_IFT511/School\_Performance\_Predictions\_random\_forest.xlsx'

dfcp.to\_excel(excel\_path, index=False)

**Code snippet:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Output snippet and Excel created snippet:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Explanation of the model:**

Random Forest Classifier Model Overview:

The Python script provided employs a Random Forest Classifier to predict school performance using a range of metrics. It starts by loading and preprocessing the data, ensuring data accuracy by removing duplicates and converting non-numeric data into suitable formats. Key metrics such as math score, English score, and graduation rate are transformed into numerical values for smooth analysis.

Model Performance Metrics:

During cross-validation, the Random Forest Classifier demonstrates commendable performance metrics. The average accuracy reaches 86%, indicating the model's proficiency in categorizing school performance. Precision and recall metrics further emphasize the model's predictive capability, with precision at 92% and recall at 90%. Additionally, the F1-score, which balances precision and recall, remains robust at 90%.

Evaluation on Test Set:

Upon evaluation on the test set, the model maintains a high accuracy of 88.24%. The detailed classification report provides insights into the model's performance for each performance category. Specifically, the model achieves flawless precision for low-performing schools (label 0), suggesting accurate predictions in this category. However, the recall for low-performing schools is relatively lower at 60%, indicating that the model misses a significant portion of actual low-performing schools. Conversely, for high-performing schools (label 1), the model achieves high precision at 86%, along with perfect recall, indicating effective identification of high-performing schools.

Predictions on Entire Dataset:

The model's predictions across the entire dataset reveal classifications of schools as either high-performing or low-performing based on learned patterns. These predictions provide valuable insights into each school's performance category, assisting stakeholders in decision-making processes.

Conclusion:

In summary, the Random Forest Classifier demonstrates robust predictive performance in school performance categorization. While exhibiting high accuracy and precision, there is an opportunity to enhance recall, particularly in identifying low-performing schools. Further refinement and validation of the model could enhance its effectiveness in practical scenarios.

**Reasons for model rebuilding and performance improvements:**

When comparing the outcomes of applying the Gradient Boosting Classifier and the Random Forest Classifier to the same school performance dataset, we can assess their effectiveness based on several criteria.

**For the Gradient Boosting Classifier:**

* During cross-validation, it demonstrated high accuracy (89%), precision (90%), recall (97%), and an F1-score (93%).
* However, its accuracy notably decreased on the test set to 76.47%. The detailed classification report highlighted perfect precision but low recall for predicting non-high-performing schools (label 0), resulting in a low F1-score for this category. In contrast, recall for high-performing schools (label 1) was perfect, with reasonably high precision.

For the Random Forest Classifier:

* Cross-validation metrics were slightly lower than Gradient Boosting in accuracy (86%), but still high for precision (92%), recall (90%), and F1-score (90%).
* Yet, its accuracy significantly improved on the test set to 88.24%. The classification report indicated enhanced performance for both categories, with perfect precision for non-high-performing schools and very high precision along with perfect recall for high-performing schools.

In practice, the Random Forest model appears to generalize better on the test set, showing higher overall accuracy and better balance between precision and recall. This suggests it may be a more dependable model for this dataset. Although cross-validation metrics initially favored Gradient Boosting, test set evaluation revealed Random Forest's superiority when dealing with unseen data. This emphasizes the importance of considering various metrics and evaluations on different data splits to fully grasp a model's performance.

These findings underscore the significance of model selection in machine learning. While Gradient Boosting may excel under certain circumstances, Random Forest's ensemble approach could offer greater robustness in others. The choice of model may depend on factors like data characteristics, class distribution, and preferred performance metrics.

To summarize, the Random Forest Classifier outperformed the initial Gradient Boosting Classifier on unseen data, which is pivotal for real-world applications where generalization is crucial.