GST PREDICTIVE ANALYTICS:REPORT

October 13, 2024

Team Members

Name	Email
Siddharth Chauhan	siddharth.chauhan@iiitb.ac.in
Kushal Partani	partani.kushal66@gmail.com
Yash Gupta	Yash.Gupta514@iiitb.ac.in
Pranav Bhutada	pranav.bhutada@iiitb.ac.in
Arjun Subhedar	arjun.subhedar@iiitb.ac.in

1 Preprocessing

1.1 Dealing with Null Values

- **Filtering:** We used Filter-Based feature selection technique named Missing Value Ratio. The Missing Value Ratio is a simple feature selection technique used to filter out features (columns) in a dataset that contain a high proportion of missing values. It is a threshold-based method in which features with missing values exceeding a specified threshold are removed from the data set. We set the threshold at 0.3.
- **Imputation:** In this method, missing values are imputed based on the mean or median of a particular group. This approach is often used when the data is categorized, and the missing values can be filled in with the central tendency (mean/median) of the group to which the row belongs.
  Mathematical Representation (Mean):

$$\bar{x}_g = \frac{1}{n_g} \sum_{i=1}^{n_g} x_i \tag{1}$$

- **Interpolation (Polynomial of Order 2):** In this method, we fill missing values using polynomial interpolation with order 2. Polynomial interpolation constructs a polynomial of degree n that passes

through the known data points. The missing values are approximated by this polynomial.
Mathematical Representation:

$$f(x) = a_2x^2 + a_1x + a_0 \tag{2}$$

- **Interpolation (Linear):** This method uses linear interpolation, where the missing value is filled based on the values of the two adjacent data points. It assumes a straight line between two known points and fills missing values accordingly.
Mathematical Representation:

$$f(x) = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1 \tag{3}$$

Where, x_1, y_1 and x_2, y_2 are the coordinates of the known points.

1.2 Dealing with Outliers

Points which lied below the 25% quartile range and above the 75% quartile range were removed from the data set.

1.3 Normalization

We applied the standard scale to transform the features of X_train into normalized values.

2 Feature Selection

2.1 Chi-Squared Test

The Chi-Squared test is used to determine if there is a significant relationship between two categorical variables — in this case, each feature and the target variable. A higher Chi-Squared score means the feature is more strongly associated with the target.The p-value tells us whether the association is statistically significant. A lower p-value indicates that the relationship is unlikely to be due to chance.

Formula The formula for the Chi-Squared statistic (χ^2) is given by:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{4}$$

Where:

- O_i = Observed frequency in each category
- E_i = Expected frequency in each category

Explanation The Chi-Squared statistic measures the discrepancy between the observed and expected frequencies. A higher value of χ^2 indicates a greater difference between observed and expected counts, suggesting a potential relationship between the variables.

- If the calculated χ^2 falls outside a critical value range determined by the degrees of freedom and significance level, we reject the null hypothesis. This implies that the two variables are likely correlated. -

Conversely, if the χ^2 value is within the bounds, we fail to reject the null hypothesis, suggesting that the variables are independent and the observed differences may occur by chance.

2.2 Information Gain

Information Gain (IG) is a commonly used criterion for feature selection in machine learning, particularly in decision trees and other classification tasks.
Information gain is based on the concept of entropy, which is a measure of uncertainty or impurity in a dataset. The main idea behind using information gain for feature selection is that features which provide the most reduction in uncertainty about the class are considered the most informative and useful for classification. **Information Gain (IG)** is the difference between the entropy of the original dataset and the weighted entropy of the subsets after splitting on the feature:

$$IG(S, \text{feature}) = H(S) - \sum_{i=1}^{k} \frac{|S_i|}{|S|} H(S_i)$$

Where:

- S is the original dataset.
- S_i is a subset created by splitting on the feature.
- $H(S_i)$ is the entropy of subset S_i.
- $\frac{|S_i|}{|S|}$ is the proportion of examples in subset S_i.

3 Models:

3.1 Random Forest Classifier

	Accuracy	Precision	Recall	F1 Score	AUC-ROC	Balanced Accuracy
Desc1	0.9688	0.8047	0.8833	0.8421	0.9892	0.9305

Table 1: Validation accuracy and loss scores

Now created binary focal loss for integrating in logistic regression.The Focal loss is a variant of the binary cross entropy loss that addresses the issue of class imbalance with the standard cross entropy loss by down-weighting the contribution of easy examples enabling learning of harder examples.

$$\text{LBCE}(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right)$$

$$LF(p_t) = \alpha(1-p_t)^{\gamma} \cdot \text{LBCE}(p, y)$$

Metrics are:

Metric	Value
Accuracy	0.9687
Precision	0.8284
Recall	0.8433
F1 Score	0.8358
AUC-ROC	0.9893
Balanced Accuracy	0.9126

Table 2: Performance Metrics

```
Confusion Matrix:
 [[232724   4310]
 [  3867  20811]]
```

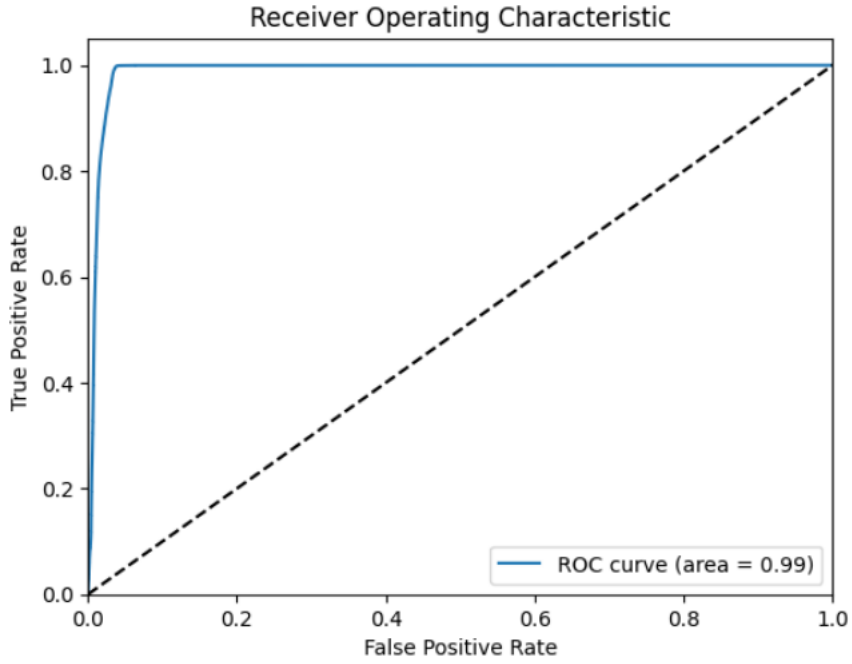


Figure 1: ROC Curve

Random Forest:

```
rforest = RandomForestClassifier(n_estimators=1500, random_state=42)
rforest.fit(X_train, y_train)
```

Metric	Value
Accuracy	0.9772
Precision	0.8464
Recall	0.9263
F1 Score	0.8846
AUC-ROC	0.9941
Balanced Accuracy	0.9544

Table 3: Random Forest Performance Metrics

```
Confusion Matrix:
 [[232886   4148]
 [  1819  22859]]
```

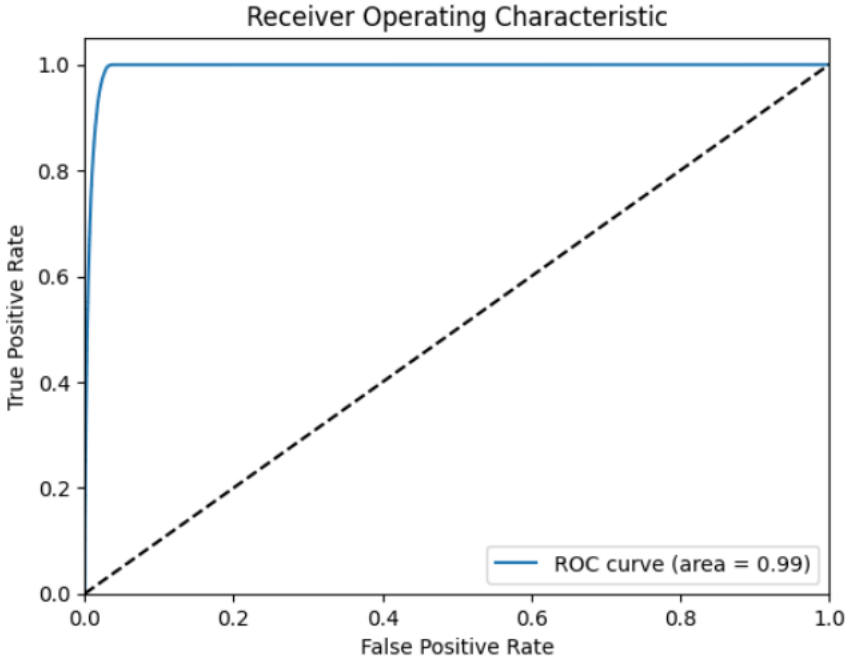


Figure 2: ROC Curve

4 Method:2

Preprocessing

Merged X_train and Y_train on basis of ID.

```
data = pd.merge(train_x, test_x, on='ID')
data.head()
```

Now used Filter-Based Feature Selection Technique named Missing Value Ratio . The Missing Value Ratio is a simple feature selection technique used to filter out features (columns) in a dataset that contain a high proportion of missing values. It is a threshold-based method where features with missing values exceeding a specified threshold are removed from the dataset. Setted the threshold to 0.3.

Method 1: Imputing with Group Mean/Median

Description: In this method, missing values are imputed based on the mean or median of a particular group. This approach is often used when the data is categorized, and the missing values can be filled with the central tendency (mean/median) of the group to which the row belongs.

Mathematical Representation (Mean):

$$\bar{x}_g = \frac{1}{n_g} \sum_{i=1}^{n_g} x_i \tag{5}$$

Where $\bar{x}_g$ is the mean of group g, and n_g is the number of values in group g.

Python Implementation:

```
df['column'].fillna(df.groupby('group')['column'].transform('mean'))
```

Alternatively, using median:

```
df['column'].fillna(df.groupby('group')['column'].transform('median'))
```

We grouped by column targ

Method 2: Interpolation - Polynomial Order 2

Description: In this method, we fill missing values using polynomial interpolation with order 2. Polynomial interpolation constructs a polynomial of degree n that passes through the known data points. The missing values are approximated by this polynomial.

Mathematical Representation:

$$f(x) = a_2 x^2 + a_1 x + a_0 \tag{6}$$

Where a_2, a_1, a_0 are the coefficients determined from the existing data points.

Python Implementation:

```
df['column'].interpolate(method='polynomial', order=2)
```

Method 3: Interpolation - Linear

Description: This method uses linear interpolation, where the missing value is filled based on the values of the two adjacent data points. It assumes a straight line between two known points and fills missing values accordingly.

Mathematical Representation:

$$f(x) = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1 \tag{7}$$

Where x_1, y_1 and x_2, y_2 are the coordinates of the known points.

Python Implementation:

```
df['column'].interpolate(method='linear')
```

Now used Chi-Squared Test to select best features.

Python Implementation:

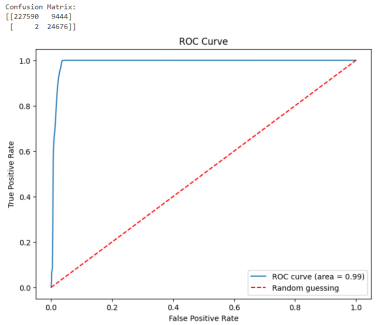
```
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler

# X: Categorical feature DataFrame, y: Target
X = data_train.drop('target', axis=1)
y = data_train['target']

# Chi-square requires non-negative values, so we scale X to [0, 1]
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Perform Chi-Squared test
chi_scores = chi2(X_scaled, y)

# Create a DataFrame to display Chi-Squared scores and p-values
chi2_df = pd.DataFrame({'Feature': X.columns, 'Chi-Squared Score': chi_scores[0],
'p-value': chi_scores[1]})

# Sort features by Chi-Squared score
chi2_df.sort_values(by='Chi-Squared Score', ascending=False, inplace=True)
```

Removed features with low relevance based on chi-sqaured test. Now applied different models to three cases.

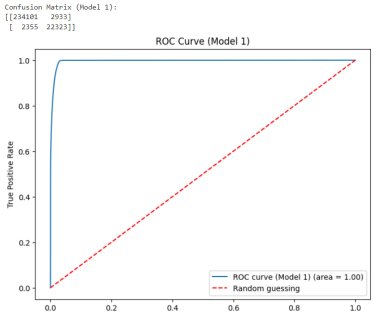
Logistic Regression

```
logreg = LogisticRegression(class_weight='balanced', max_iter=1000)
# max_iter is set high to ensure convergence
logreg1 = LogisticRegression(class_weight='balanced', max_iter=1000)
logreg2 = LogisticRegression(class_weight='balanced', max_iter=1000)
```

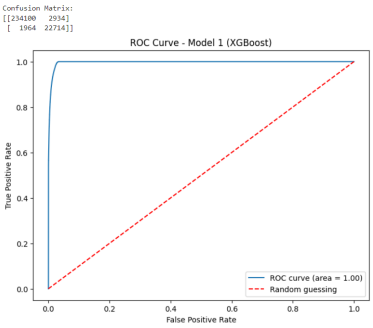
5 Appendicks


Confusion Matrix:
[[227590 9444]
[2 24676]]
ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)
Random guessing

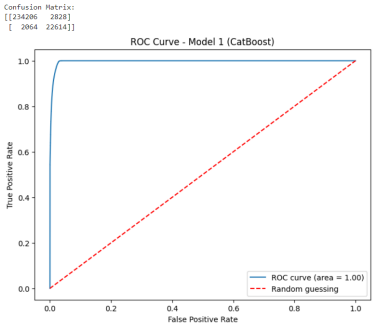
(a)


Confusion Matrix (Model 1):
[[234101 2933]
[2355 22323]]
ROC Curve (Model 1)
True Positive Rate
False Positive Rate
ROC curve (Model 1) (area = 1.00)
Random guessing

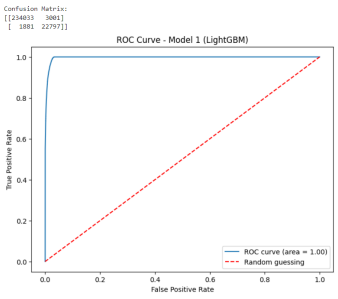
(b)


Confusion Matrix:
[[234100 2934]
[1964 22714]]
ROC Curve - Model 1 (XGBoost)
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
Random guessing

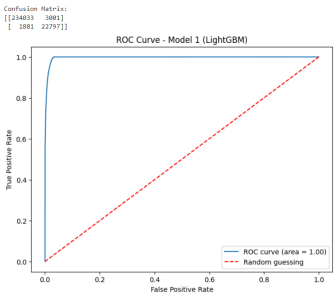
(c)


Confusion Matrix:
[[234206 2828]
[2064 22614]]
ROC Curve - Model 1 (CatBoost)
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
Random guessing

(d)


Confusion Matrix:
[[234033 3001]
[1881 22797]]
ROC Curve - Model 1 (LightGBM)
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
Random guessing

(e)


Confusion Matrix:
[[234033 3001]
[1881 22797]]
ROC Curve - Model 1 (LightGBM)
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
Random guessing

(f)

Table 4: Comparison of Model Performance with Different Preprocessing Methods

Model	**Accuracy**	**Precision**	**F1 Score**	**ROC-AUC**	**Recall**	**Balanced Accuracy**	**Preprocessing Method**
Logistic	0.9639	0.7232	0.8393	0.9897	0.9999	0.9800	Group Imputing
Random Forest	0.9798	0.8839	0.8941	0.9963	0.9046	0.9461	Group Imputing
XGBoost	0.9813	0.8856	0.9027	0.9970	0.9204	0.9540	Group Imputing
CatBoost	0.9813	0.8888	0.9024	0.9970	0.9164	0.9522	Group Imputing
LightGBM	0.9813	0.8837	0.9033	0.9970	0.9238	0.9556	Group Imputing
Logistic	0.9643	0.7254	0.8408	0.9890	0.9998	0.9802	Interpolation-Polynomial
Random Forest	0.9738	0.8359	0.8662	0.9927	0.8988	0.9402	Interpolation-Polynomial
XGBoost	0.9755	0.8315	0.8775	0.9939	0.9289	0.9547	Interpolation-Polynomial
CatBoost	0.9755	0.8338	0.8771	0.9938	0.9250	0.9529	Interpolation-Polynomial
LightGBM	0.9757	0.8273	0.8792	0.9939	0.9380	0.9588	Interpolation-Polynomial
Logistic	0.9688	0.8047	0.8421	0.9892	0.8833	0.9305	Mean Imputing
AdaBoost	0.9755	0.8241	0.8784	0.9933	0.9404	0.9598	Mean Imputing
Random Forest	0.9771	0.8470	0.8840	0.9941	0.9245	0.9535	Mean Imputing
XGBoost	0.9780	0.8468	0.8893	0.9949	0.9362	0.9593	Mean Imputing
Stacking Classifier	0.9784	0.8485	0.8911	0.9948	0.9382	0.9604	Mean Imputing
Voting Classifier	0.9784	0.8485	0.8911	0.9947	0.9382	0.9604	Mean Imputing
CatBoost	0.9784	0.8485	0.8911	0.9948	0.9382	0.9604	Mean Imputing