GST ANALYTICS PROJECT REPORT

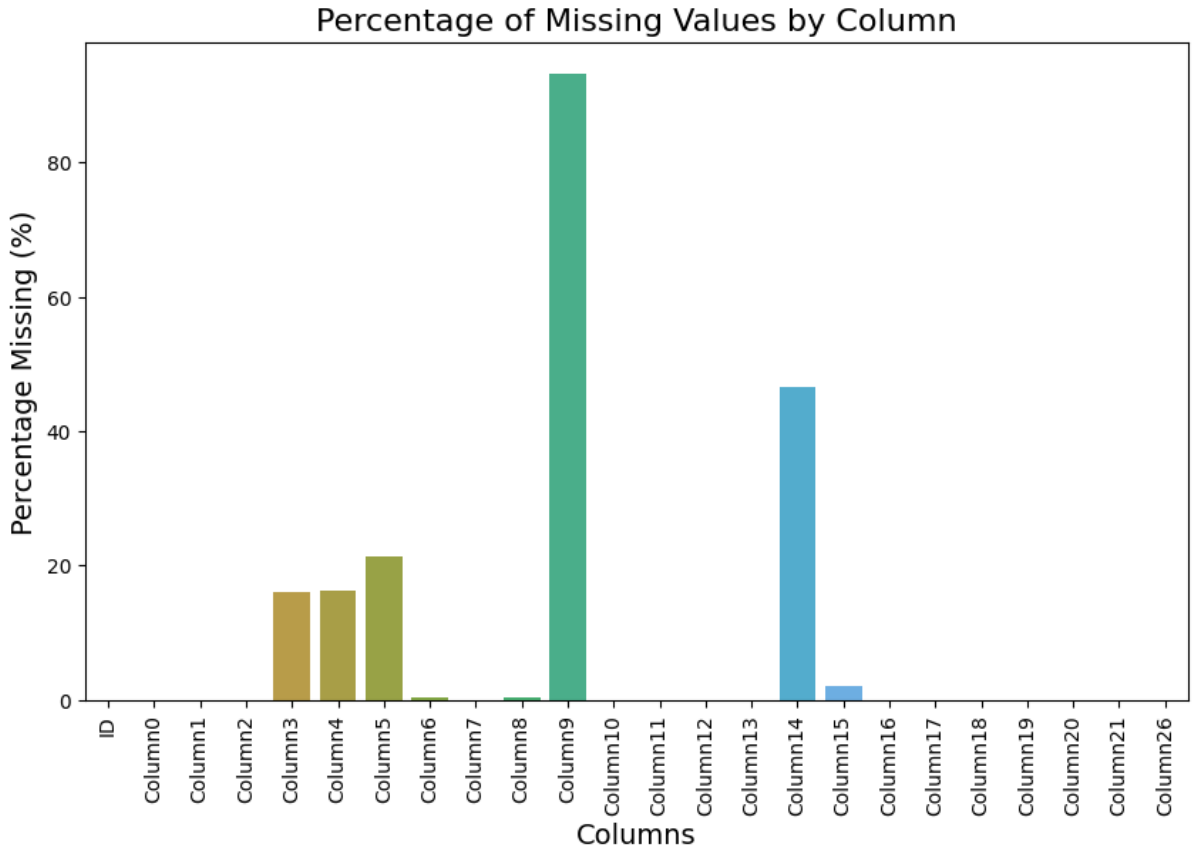
Anirvan Roy

GSTN_745
Team ID

8197836471
Anirvanroy44518@gmail.com

Step 1 of 3 : Data Preprocessing

In the initial step of the project, I began by working with the pre-split training and test datasets. A key task was identifying any missing (NaN) values within the features. After a thorough analysis of the dataset, I found that several features contained missing data, with varying amounts across the columns. Based on the characteristics of each feature, I decided to apply different imputation techniques. These included mean imputation for numerical features, mode imputation for categorical variables, and in some cases, filling missing values with a constant if it aligned with the feature's nature. A visual representation was also generated to assess the extent of missing values across features.



For the next step, I focused on addressing columns with a high percentage of missing values, starting with Column 9. Upon inspecting the NaN values, it became evident that Column 9 had approximately 90% missing data, making it unsuitable for effective imputation. Retaining such a feature could potentially introduce noise and distort the model's learning process. Therefore, I made the decision to drop Column 9 entirely from the dataset. This approach aligns with best practices, where features with an overwhelming amount of missing

data are often more detrimental than beneficial, as they lack sufficient information to contribute meaningfully to the model.

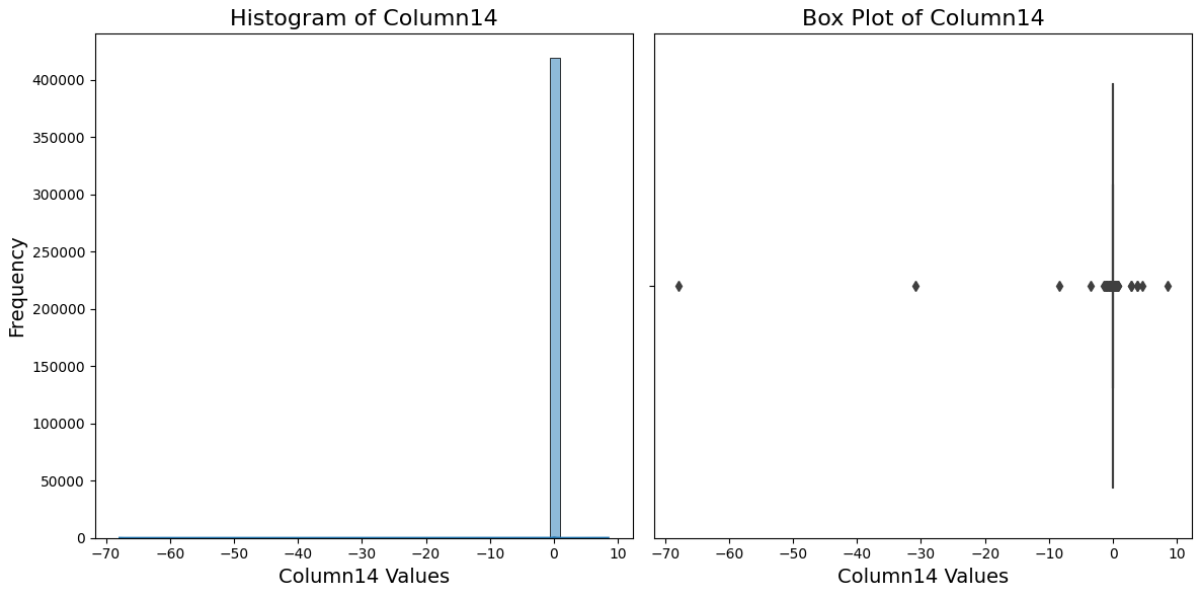
Next, I turned my attention to Column 14, which also had a significant number of missing values. Instead of dropping it outright, I decided to investigate the feature further to see if it could be salvaged through imputation. The first step involved checking for correlations between Column 14 and other features in the dataset. If a strong correlation existed, it could have provided a basis for imputing the missing values based on those correlated features. However, after calculating the correlation matrix, I found that Column 14 had weak correlations with the other features, offering no clear pathway for correlation-based imputation.

Correlation of other columns with Column14:

Column14	1.000000
Column12	0.001638
Column13	0.001405
Column11	0.001282
Column10	0.000963
Column18	0.000413
Column8	0.000222
Column20	0.000214
Column19	0.000205
Column21	0.000108
Column16	0.000038
Column26	0.000036
Column17	0.000036
Column7	0.000030
Column5	0.000010
Column15	-0.000007
Column3	-0.000826
Column4	-0.000857
Column1	-0.001666
Column2	-0.001744
Column6	-0.005014
Column0	-0.017706

Name: Column14, dtype: float64

Given that no strong correlations were identified, I then explored the distribution of values in Column 14 by plotting both a histogram and a box plot. These visualizations revealed that the vast majority of values in the column were 0, with no significant outliers present. Based on this insight, I decided to impute all the missing values in Column 14 with 0, as it was the most frequent and natural value for this feature. This approach ensured that the imputed values aligned with the existing data distribution, minimizing the risk of introducing bias or distorting the feature's overall behavior. By maintaining Column 14 with appropriate imputations, I preserved a potentially useful feature without compromising data integrity.

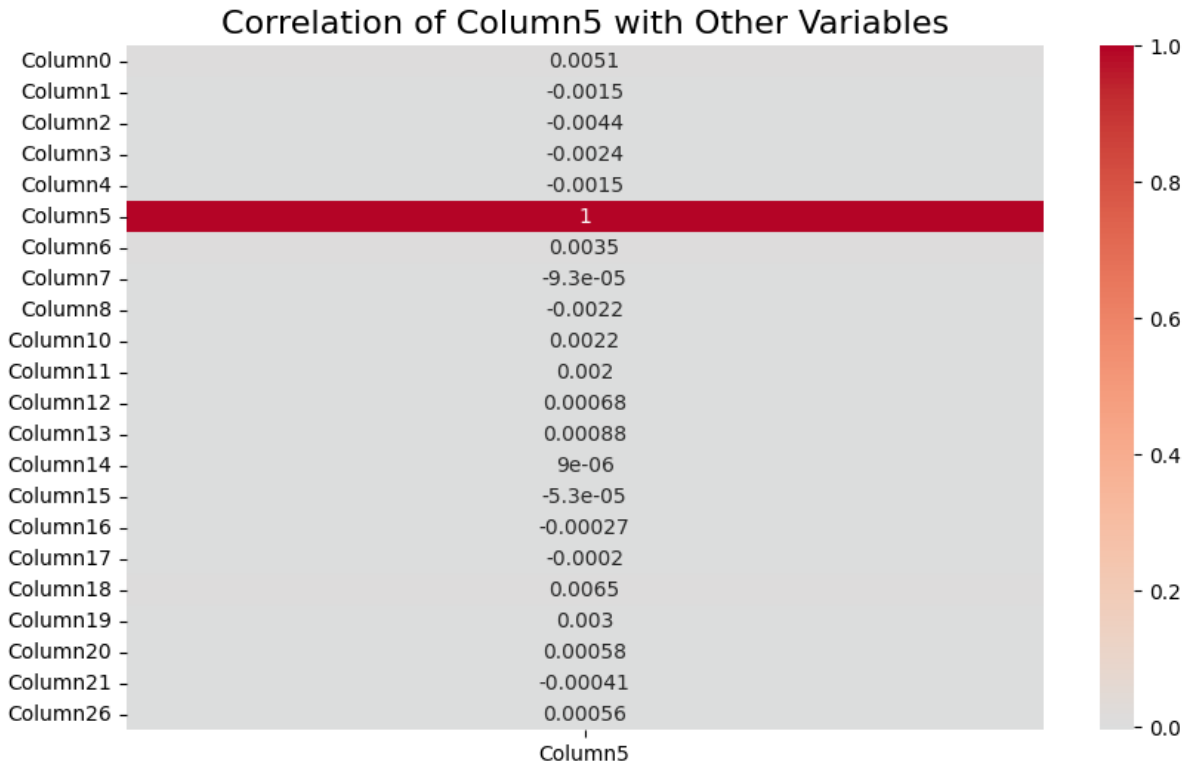


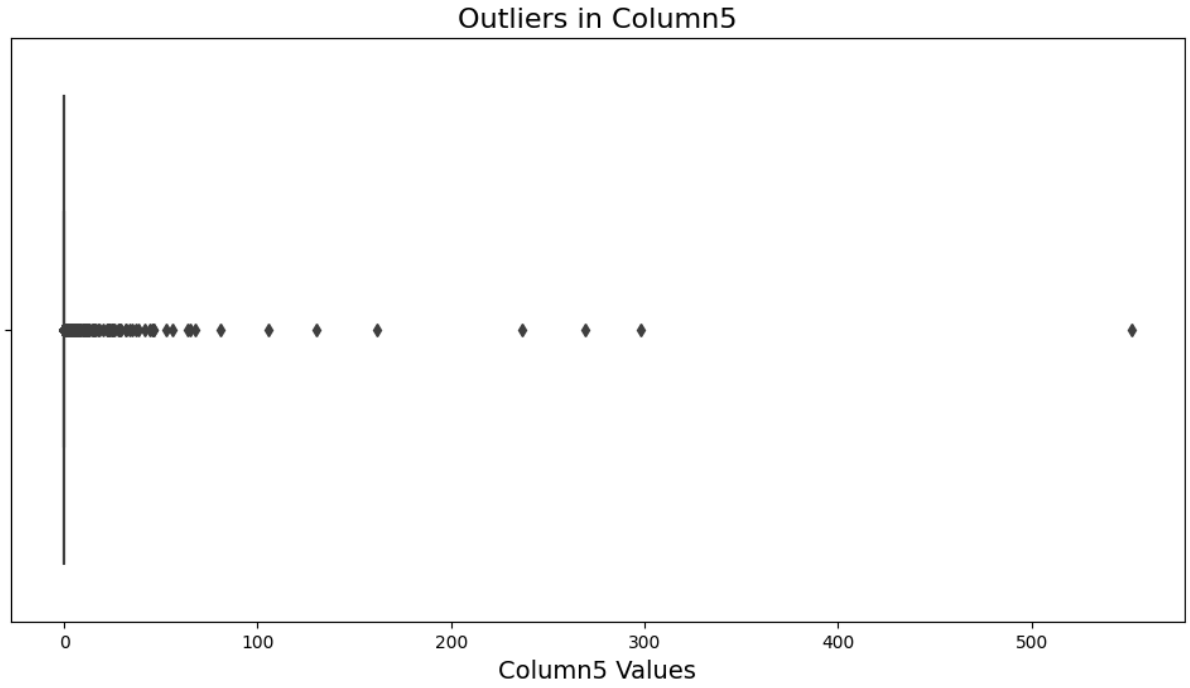
Next, I addressed Column 5, which, like Column 14, contained missing values. My first step was to check for any potential correlations with other features to explore possible imputation strategies based on related data. However, after reviewing the correlation matrix, it was evident that Column 5 did not exhibit any significant correlation with other columns in the dataset. This ruled out correlation-based imputation, prompting me to investigate the feature's distribution more closely.

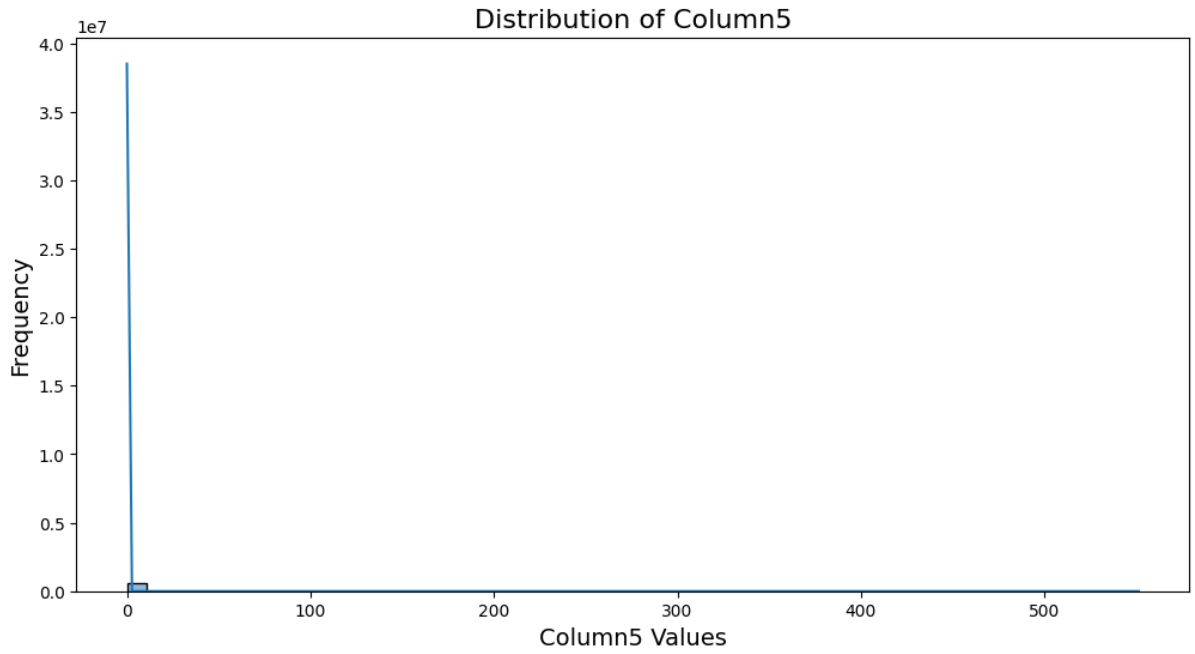
To gain insights into the behavior of Column 5, I plotted a histogram and a box plot. The histogram provided a visual summary of the distribution, while the box plot helped in identifying any extreme outliers that might skew the data. Upon analysis, it became apparent that some values in Column 5 were unusually large, particularly those exceeding 100. These values were likely outliers and could distort any statistical calculations such as the mean, which would impact the imputation process.

Considering this, I opted for a strategy that balanced the feature's overall distribution while mitigating the effect of the outliers. Specifically, I calculated the mean of Column 5 but excluded all values greater than 100 to prevent these extreme values from artificially inflating the mean. This adjusted mean better reflected the central tendency of the majority of the data points, offering a more robust imputation method.

I then used this adjusted mean to fill in all the missing values (NaNs) in Column 5. This approach ensured that the imputed values aligned with the feature's natural distribution without being skewed by outliers. By treating the missing data this way, I preserved the integrity of Column 5, allowing it to contribute meaningfully to the model without introducing noise or bias.







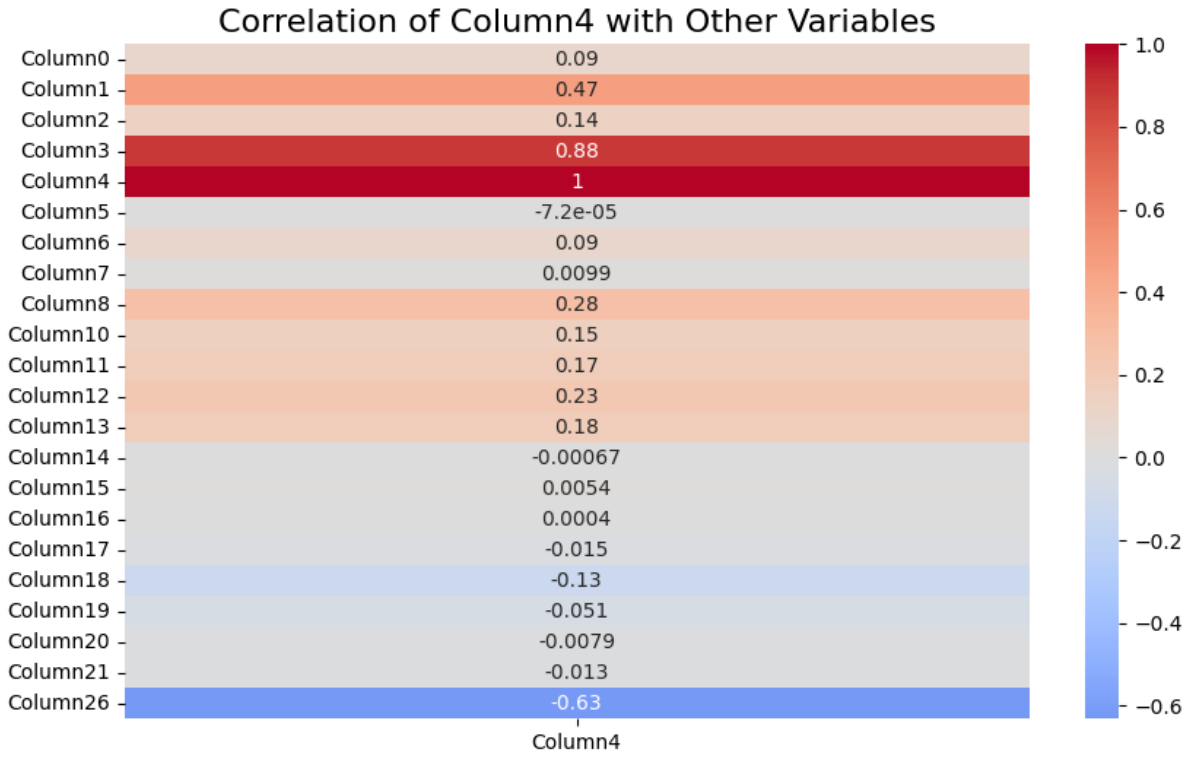
For the next step, I focused on handling the missing values in Column 4. During my initial analysis of the dataset's correlations, I discovered that Column 4 and Column 3 were highly correlated, with a correlation coefficient of 0.88. This strong linear relationship between the two features presented an opportunity to use a more sophisticated method for imputing the missing values in Column 4, as it allowed for a predictive imputation approach.

Given the strength of the correlation, I decided to implement a linear regression model to predict the missing values in Column 4 based on the available values in Column 3. The logic behind this approach is straightforward: since Column 4 and Column 3 have a nearly linear relationship, I could use the values in Column 3 to make accurate predictions about what the missing values in Column 4 would likely be. This not only fills the NaNs but does so in a way that preserves the underlying relationships within the dataset, which is crucial for maintaining model accuracy.

To achieve this, I first filtered the data to include only the rows where both Column 4 and Column 3 had non-missing values. Using this subset, I trained a linear regression model with Column 3 as the independent variable and Column 4 as the dependent variable. Once the model was trained, I used it to predict the missing values in Column 4 based on the

corresponding values from Column 3. This method leveraged the strength of the correlation, ensuring that the imputed values were in line with the natural trend between the two features.

By applying linear regression for imputation, I was able to maintain the integrity of both columns without introducing arbitrary or biased values. This approach helps in capturing the relationship between the features, making the data more suitable for downstream modeling tasks, and reducing the risk of data distortion.



Following the successful imputation of missing values in Column 4 using linear regression, I applied the same approach to Column 3. Since Columns 3 and 4 exhibited a high correlation of 0.88, the inverse relationship allowed me to use the values in Column 4 to predict the missing values in Column 3. The strong correlation provided a solid foundation for employing linear regression once again to maintain data consistency and integrity.

To begin, I created a filtered subset of the data where both Columns 3 and 4 had non-missing values. This subset served as the training data for the linear regression model. In this case, Column 4 was treated as the independent variable, while Column 3 acted as the dependent variable. By leveraging the previously established relationship between these columns, I trained the linear regression model to predict missing values in Column 3 based on the available data in Column 4.

In the next step, I addressed the missing values in Column 15. Unlike some of the other columns, Column 15 did not display any significant correlations with other features, as confirmed by analyzing the correlation matrix. Without any clear relationships to leverage for predictive imputation, I shifted focus to understanding the distribution of values within the column itself.

To gain insights into the data distribution and identify any potential outliers, I plotted both a box plot and a histogram for Column 15. The box plot was especially useful in highlighting outliers, which can skew statistical measures like the mean. Upon reviewing the box plot, I noticed that there were several extreme values, particularly those below -10, which appeared as clear outliers.

These outliers had the potential to distort the mean calculation, making it an unreliable measure for imputation if included. To ensure that the mean more accurately reflected the central tendency of the non-outlier data, I excluded all values less than -10 from the calculation. This approach allowed me to compute an adjusted mean that better represented the majority of the data while minimizing the influence of extreme negative values.

I then proceeded to impute the missing values in Column 15 using this adjusted mean. By excluding the extreme outliers from the calculation, I ensured that the imputed values would align with the natural data distribution, offering a more accurate and reliable fill for the missing entries. This approach maintained the integrity of Column 15 while avoiding the negative impact that outliers could have had on both the imputed values and the overall model performance.

This step reinforced the importance of handling outliers appropriately during the imputation process, as failing to account for them could lead to biased or misleading results in subsequent modeling.

For Columns 8 and 6, I applied the same imputation strategy used for Column 15. I began by analyzing both columns using box plots to identify outliers that could distort the imputation process. Once the outliers were flagged, I excluded extreme values from the mean calculation, specifically values below certain thresholds, ensuring that the imputed values would be more representative of the general distribution. This method prevented skewness and preserved the overall integrity of the dataset.

Next, I turned my attention to Column 0. Upon examining the data, I found no significant correlations between Column 0 and any other features, ruling out correlation-based imputation. To better understand its nature, I plotted a box plot to visualize the distribution. Based on the pattern and the values, I hypothesized that Column 0 might be categorical, as it exhibited characteristics of distinct groupings rather than a continuous distribution. Given this, I decided to take a more quantile-based approach to imputation, as mean imputation might not accurately reflect the structure of a categorical feature.

I calculated the mean based on the 75th quantile of the data. This approach ensured that I was imputing the missing values with a value that leaned toward the higher end of the distribution, which seemed appropriate based on the visual analysis of the box plot. The 75th quantile often captures the upper segment of a distribution, which, in categorical data, can provide a reasonable representation of frequent or common values while avoiding the distortion caused by extreme outliers or rare categories.

By imputing the missing values in Column 0 with the mean of the 75th quantile, I was able to maintain the structure and integrity of the feature, ensuring it remained useful for the model without introducing undue bias.

Step 2 of 3 : Modelling

After experimenting with multiple models, including logistic regression, random forest, XGBoost, and AdaBoost, I found that LightGBM provided the most promising results for my dataset. LightGBM was chosen as the base model due to its superior performance across multiple evaluation metrics, which demonstrated its effectiveness at handling the complexities of the data.

For the LightGBM model, the training accuracy was recorded at an impressive 0.9791, indicating that the model performed well on the training data. More importantly, the test accuracy was very close, at 0.9782, showing that the model generalizes well to unseen data and is not overfitting.

The confusion matrix on the test data showed:

True Positives: 23270

False Positives: 4297

False Negatives: 1408

True Negatives: 232737

These results highlight the model's ability to accurately distinguish between the two classes. The recall score of 0.9429 indicated that the model was able to identify 94.29% of the positive instances correctly. However, the precision was slightly lower at 0.8441, meaning that 84.41% of the predicted positives were indeed correct.

The F1 score, which balances precision and recall, was 0.8908, reflecting strong overall performance in both metrics. Lastly, the AUC-ROC score of 0.9949 demonstrated excellent model discrimination, as the model was almost perfect at distinguishing between positive and negative classes.

Given these results, LightGBM was chosen as the final model, offering both high accuracy and balanced performance across recall, precision, and AUC-ROC. This makes it a reliable choice for further model refinement and deployment.

Step 3 of 3 : Feature Engineering and Re-Modelling

Based on the feature importance analysis, I engaged in a process of feature engineering primarily through trial and error, aiming to enhance the performance of my LightGBM model. This iterative process involved creating new features that capture the relationships between existing variables, thereby providing the model with additional information to work with.

First, I introduced **Column 24**, which is the product of **Column 2** and **Column 7**. This interaction term was intended to highlight any combined effects these features might have on the target variable. Next, I created **Column 25**, which is derived from the product of **Column 7** and **Column 8**. This feature aimed to explore further interactions between key variables that could contribute to predictive power.

In addition to these multiplicative features, I created **Column 26** using a conditional statement: if **Column 0** is greater than 0, then it is multiplied by **Column 7**; otherwise, it retains the value from **Column 7**. This conditional approach allows for capturing the influence of **Column 0** when it is positive while maintaining a fallback value when it is not.

Similarly, I constructed **Column 27**, where the multiplication occurs between **Column 0** and **Column 14** under the same conditions, capturing interactions that might be particularly relevant in the context of the target variable.

Furthermore, I generated **Column 28** as the sum of **Column 12** and **Column 2**, which aimed to incorporate additive effects from these features. Lastly, I created **Column 29** by summing **Column 14** and **Column 12**, seeking to combine these features to reflect their collective impact.

Through this targeted feature engineering approach, I aimed to improve model performance by capturing more nuanced relationships within the dataset, which could lead to better predictive accuracy and a more robust final model. Each new feature was carefully considered, allowing the LightGBM model to leverage a richer set of data for decision-making.

After implementing the feature engineering enhancements, I retrained my LightGBM model to assess the impact of the newly created features on its performance. The results showed a significant improvement, with a test accuracy of 0.9828, indicating that the model correctly predicted the target variable for approximately 98.28% of the test cases.

The confusion matrix for the test data revealed the following:

- True Positives: 9055
- False Positives: 2285
- False Negatives: 1100
- True Negatives: 183877

These results demonstrate that the model was effective at distinguishing between the positive and negative classes, with the majority of predictions being accurate. The recall score was recorded at 0.8917, indicating that the model successfully identified 89.17% of the actual positive instances. However, the precision was slightly lower at 0.7985, meaning that about 79.85% of the positive predictions made by the model were correct.

The F1 score of 0.8425 reflects a balance between precision and recall, suggesting that while there is room for improvement, the model performs reasonably well in terms of both metrics. Finally, the AUC-ROC score of 0.9952 signifies excellent model discrimination, confirming that the LightGBM model is highly effective in distinguishing between the classes.

Overall, the retraining of the LightGBM model, enhanced by feature engineering, yielded positive results that underscore the value of refining input features to boost predictive performance. The significant improvement in accuracy and the favorable metrics highlight the importance of careful feature selection and engineering in machine learning processes.

References

Di Russo, J. (2021, December 12). Navigating The Hell of NaNs in Python - Towards Data Science. Medium. https://towardsdatascience.com/navigating-the-hell-of-nans-in-python-71b12558895b

Innovative Ways to Enhance ML Models with Feature Engineering. (n.d.). https://www.markovml.com/blog/feature-engineering-best-practices

Loukas, S., PhD. (2024, February 23). How To Perform Feature Selection for Regression Problems. Medium. https://towardsdatascience.com/how-to-perform-feature-selection-for-regression-problems-c928e527bbfa

Parsons, N. (2023, November 30). Feature engineering for fraud detection. Fennel Blog. https://fennel.ai/blog/feature-engineering-for-fraud-detection/

Peeters, J. (2021, December 14). A framework for feature engineering and machine learning pipelines. Medium. https://medium.com/manomano-tech/a-framework-for-feature-engineering-and-machine-learning-pipelines-ddb53867a420