Fraud Detection Using Supervised Learning

by Farzam Manafzadeh

Made with Gamma

About the Dataset

This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants.

Source of Simulation

This was generated using **Sparkov Data Generation | Github** tool created by Brandon Harris. This simulation was run for the duration - 1 Jan 2019 to 31 Dec 2020. The files were combined and converted into a standard format.

Made with Gamma

Data Set

1. **trans_date_trans_time:** The date and time of the transaction.
2. **cc_num:** Credit card number used in the transaction.
3. **merchant:** The name of the merchant involved in the transaction.
4. **category:** The category of the transaction (e.g., groceries, electronics).
5. **amt:** The amount of money involved in the transaction.
6. **first:** First name of the person associated with the credit card.
7. **last:** Last name of the person associated with the credit card.
8. **gender:** Gender of the person associated with the credit card.
9. **street:** Street address of the person associated with the credit card.
10. **city:** City where the person associated with the credit card resides.
11. **state:** State where the person associated with the credit card resides.
12. **zip:** ZIP code of the person associated with the credit card.
13. **lat:** Latitude coordinate of the transaction location.
14. **long:** Longitude coordinate of the transaction location.
15. **city_pop:** Population of the city where the person resides.
16. **job:** Occupation or job title of the person associated with the credit card.
17. **dob:** Date of birth of the person associated with the credit card.
18. **trans_num:** Transaction number.
19. **unix_time:** Unix timestamp of the transaction.
20. **merch_lat:** Latitude coordinate of the merchant's location.
21. **merch_long:** Longitude coordinate of the merchant's location.
22. **is_fraud:** Binary indicator (0 or 1) representing whether the transaction is fraudulent or not.

Made with Gamma

Data preprocessing

Dataset Overview:

- The dataset consists of 1,296,675 entries and 22 columns.
- Initial column names: 'trans_date_trans_time', 'cc_num', 'merchant', 'category', 'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip', 'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time', 'merch_lat', 'merch_long', 'is_fraud'.

Column Renaming:

- Renamed columns for clarity:
  - 'trans_date_trans_time' to 'transaction_time'.
  - 'cc_num' to 'credit_card_number'.
  - 'amt' to 'amount(usd)'.
  - 'trans_num' to 'transaction_id'.

Datetime Conversion:

- Converted 'transaction_time' and 'dob' columns to datetime format.
- Created a new 'time' column using the 'unix_time' column.
- Extracted the 'hour_of_day' from the 'time' column.

Data Type Conversion:

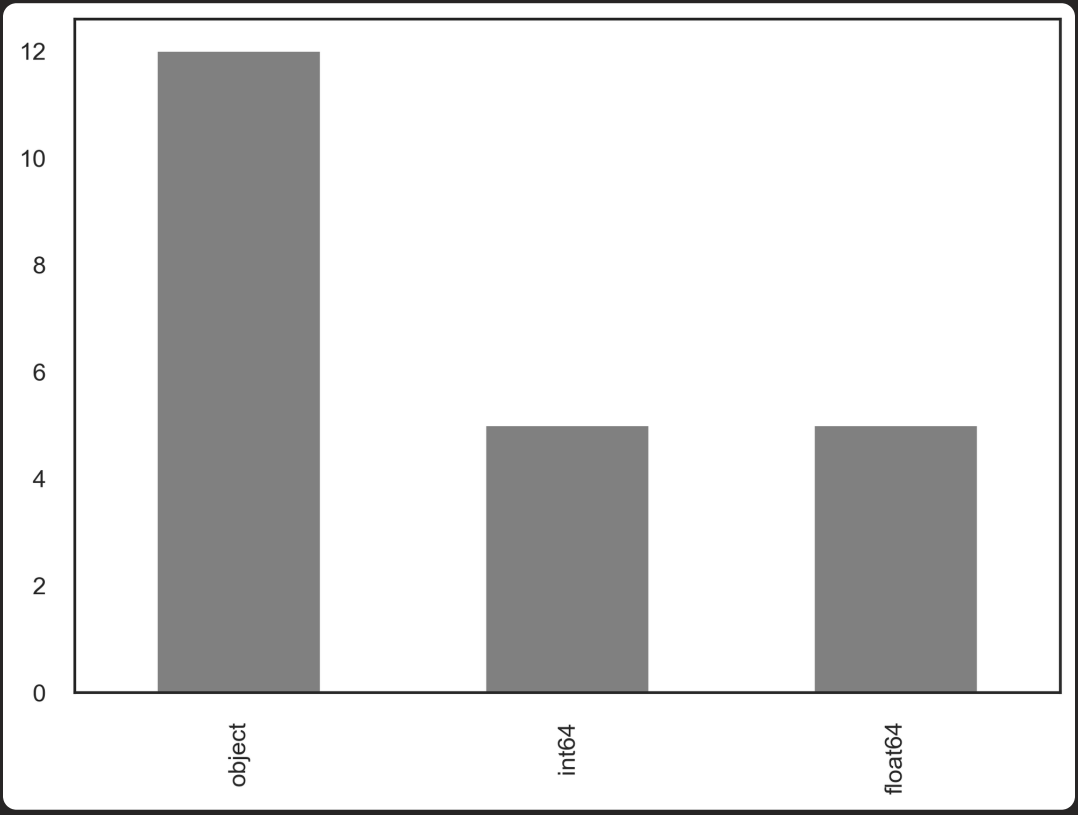
- Converted 'credit_card_number', 'hour_of_day', and 'is_fraud' to categorical data type.

Final Dataset:

- The transformed dataset includes 24 columns with updated column names and data types.
- the final data set has no Missing values.

Made with Gamma

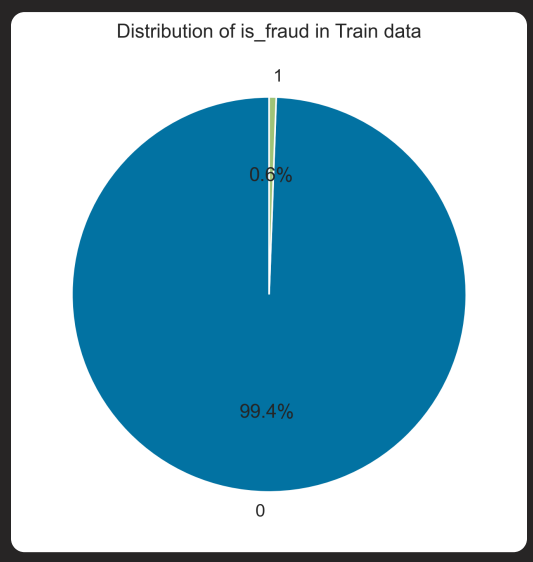
Exploratory Data Analysis

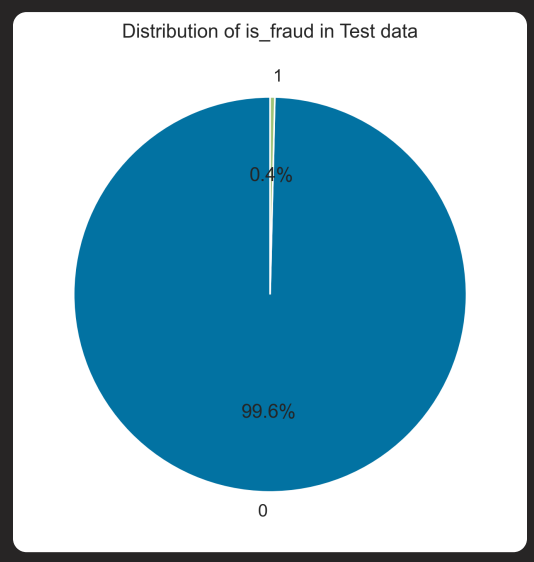


There are a lot of categorical features! So we are probably going to consider categorical encoding.

Made with Gamma

Univariate analysis

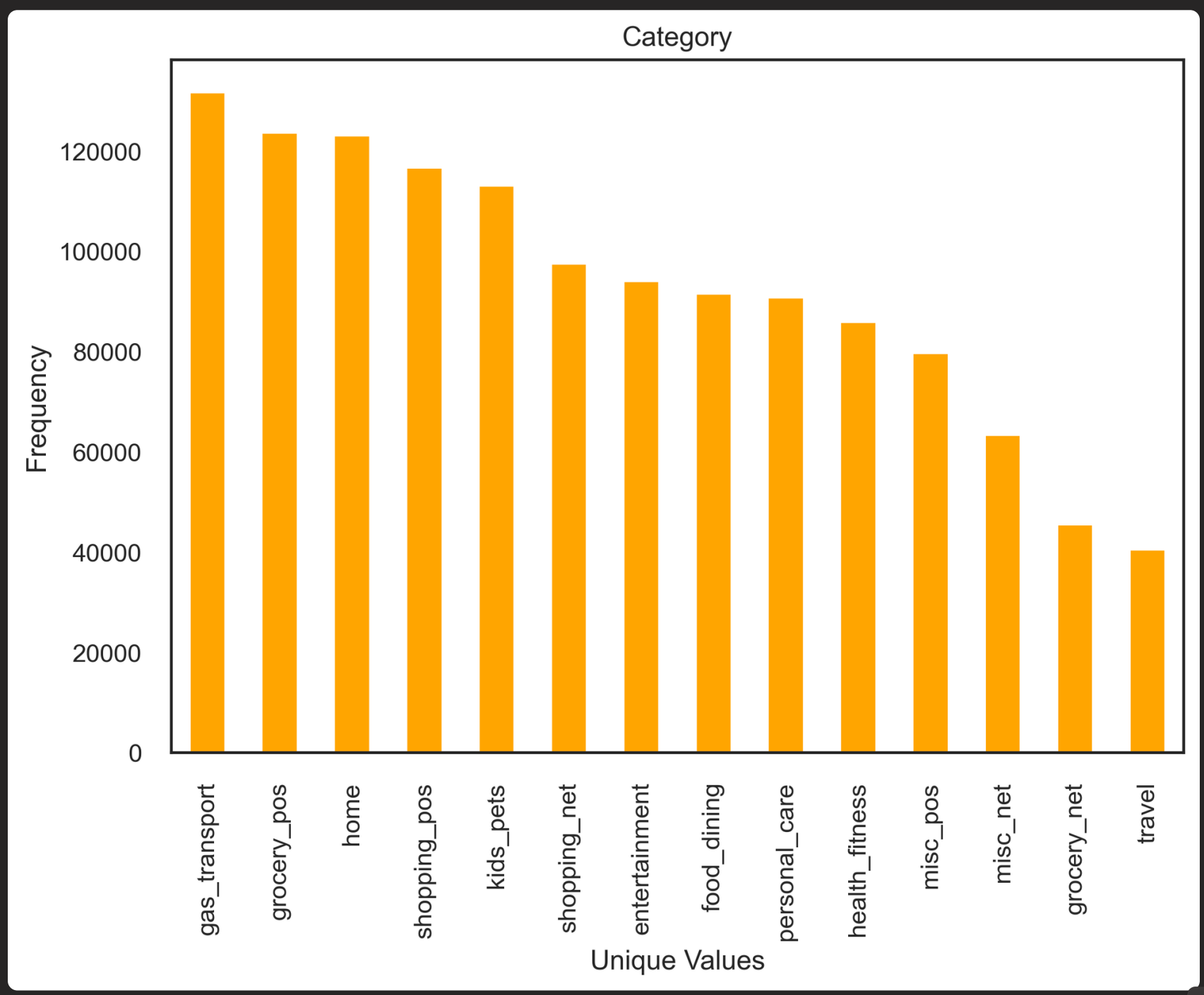




We have a severe imbalance dataset. Because of this, Oversampling/Undersampling will be considered in the model.

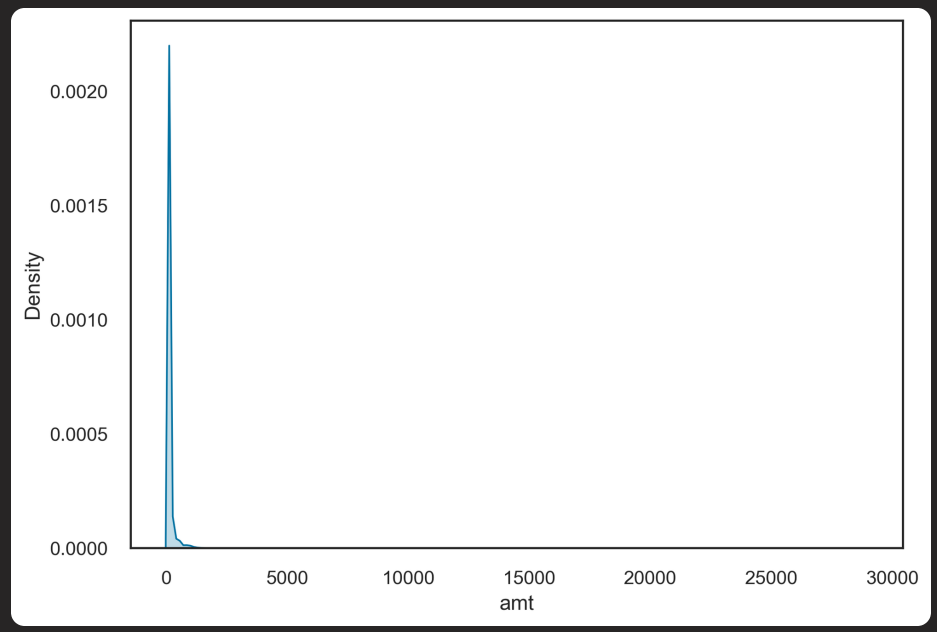
Both train and test dataset are similarly imbalanced. As such, cross-validation techniques for partitioning purposes are not required.

Categories Frequencies :

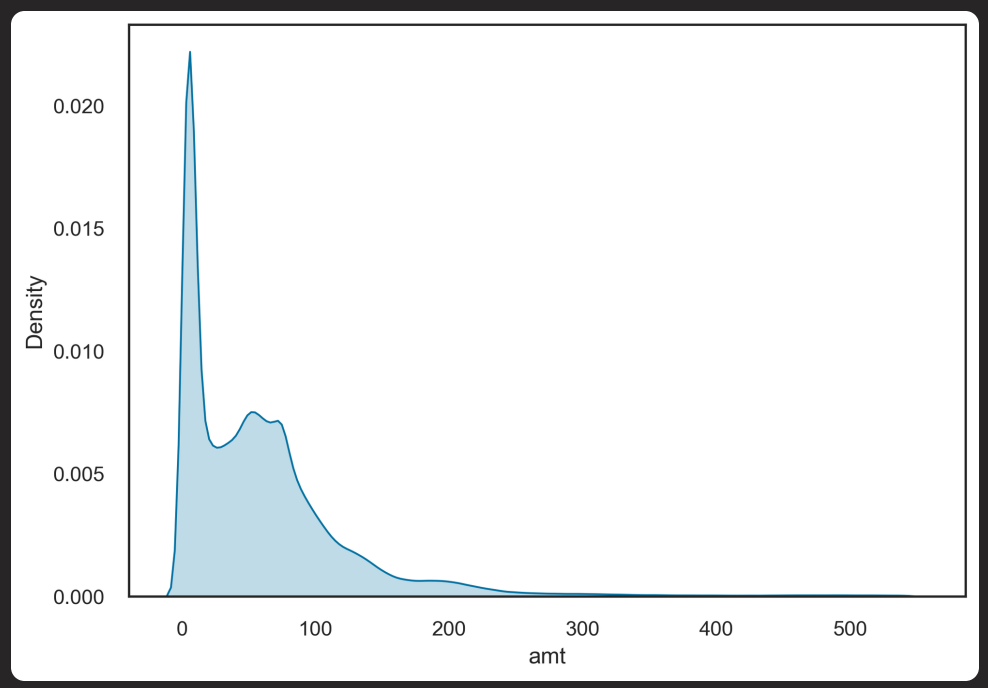


Made with Gamma

Let's check the distribution of the amount ("amt") feature.



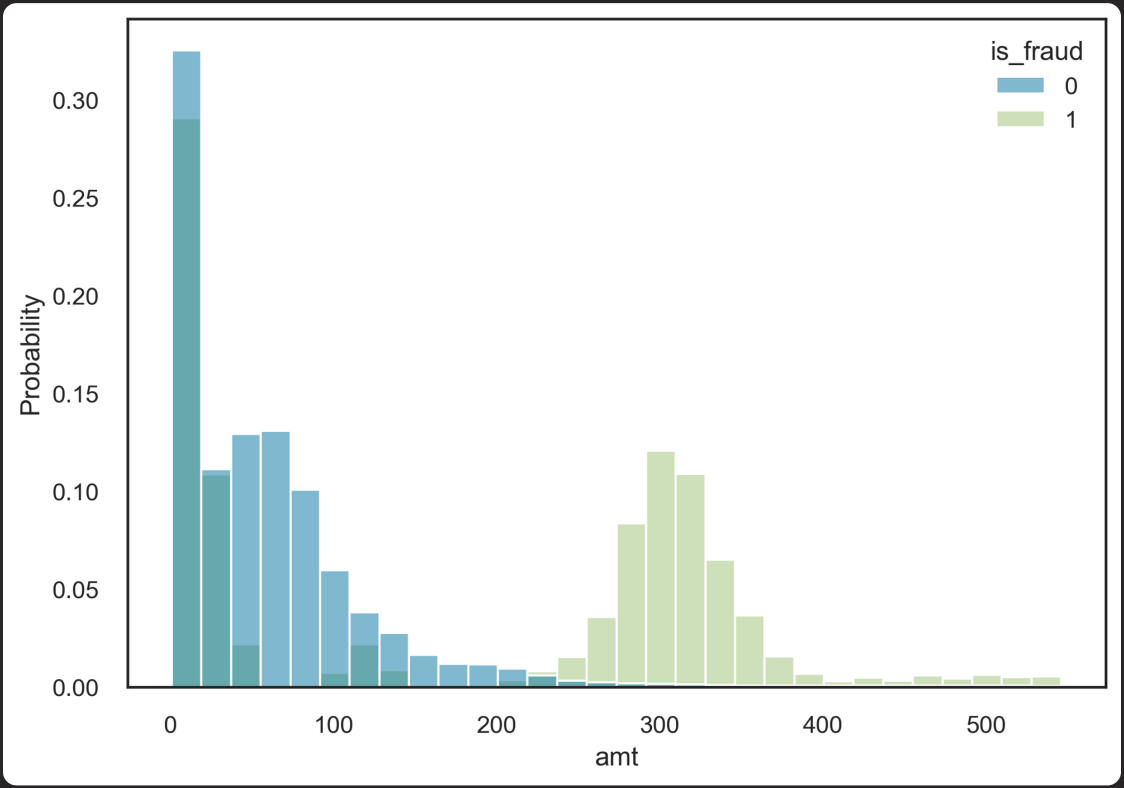
Higher values of amount is distorting the graph. So we fix it plotting only 0-99% of amount.



Made with Gamma

Multivariate analysis

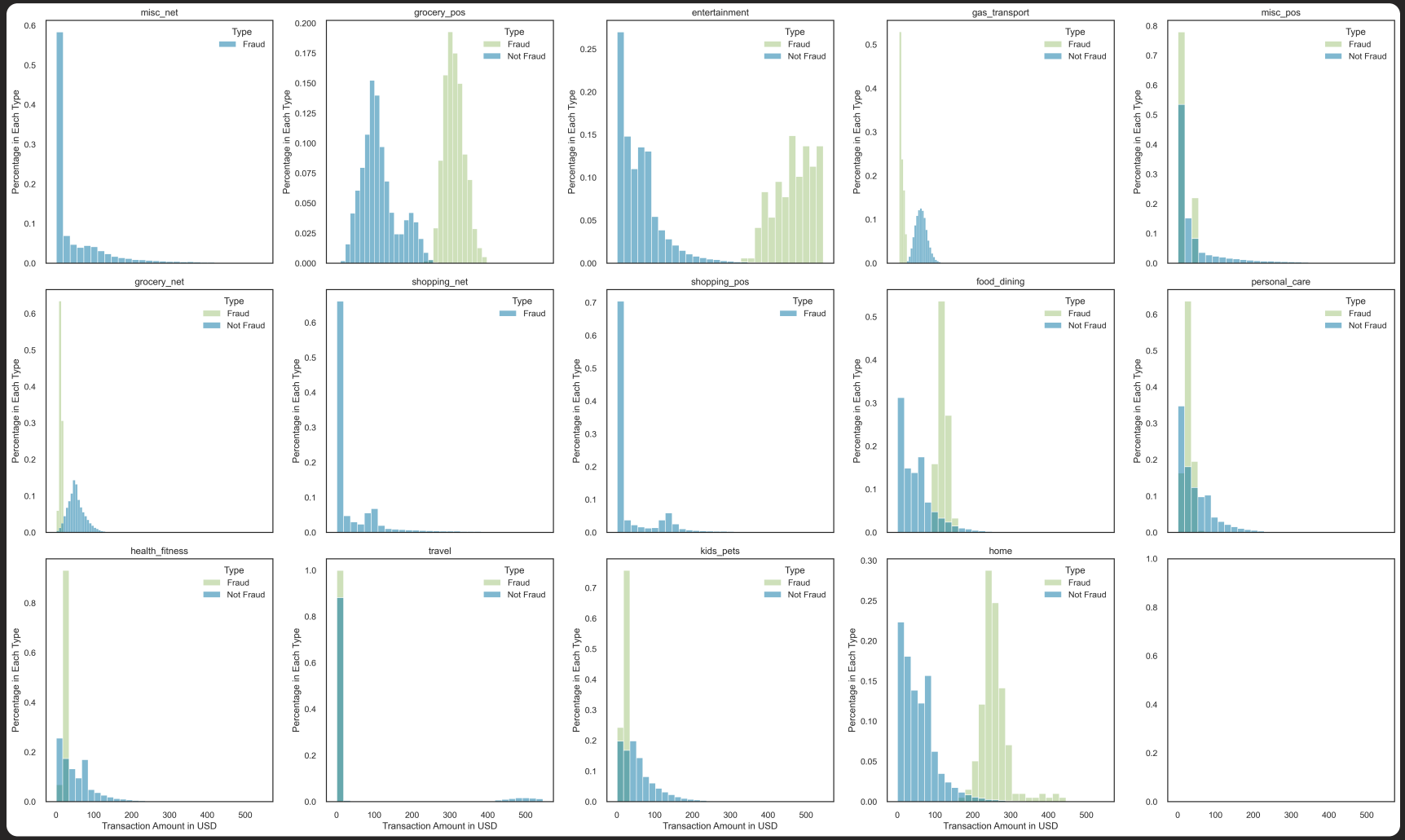
Lets se if there is any noticeable pattern between the target (is_fraud) and amount.



Notice that the probability distribution behaves differently for each value of "is_fraud".

We can expand this analysis to each transaction category

Made with Gamma



Made with Gamma

Categories

Lets dive into categories. We'll analyze which categories the frauds are more frequently. For this, we'll calculate the distribution for each category for normal transactions and then the distribution for fraudulents transactions. Then, we take the difference between the two distributions. This difference shows us the categories that are most predominants to have fraud

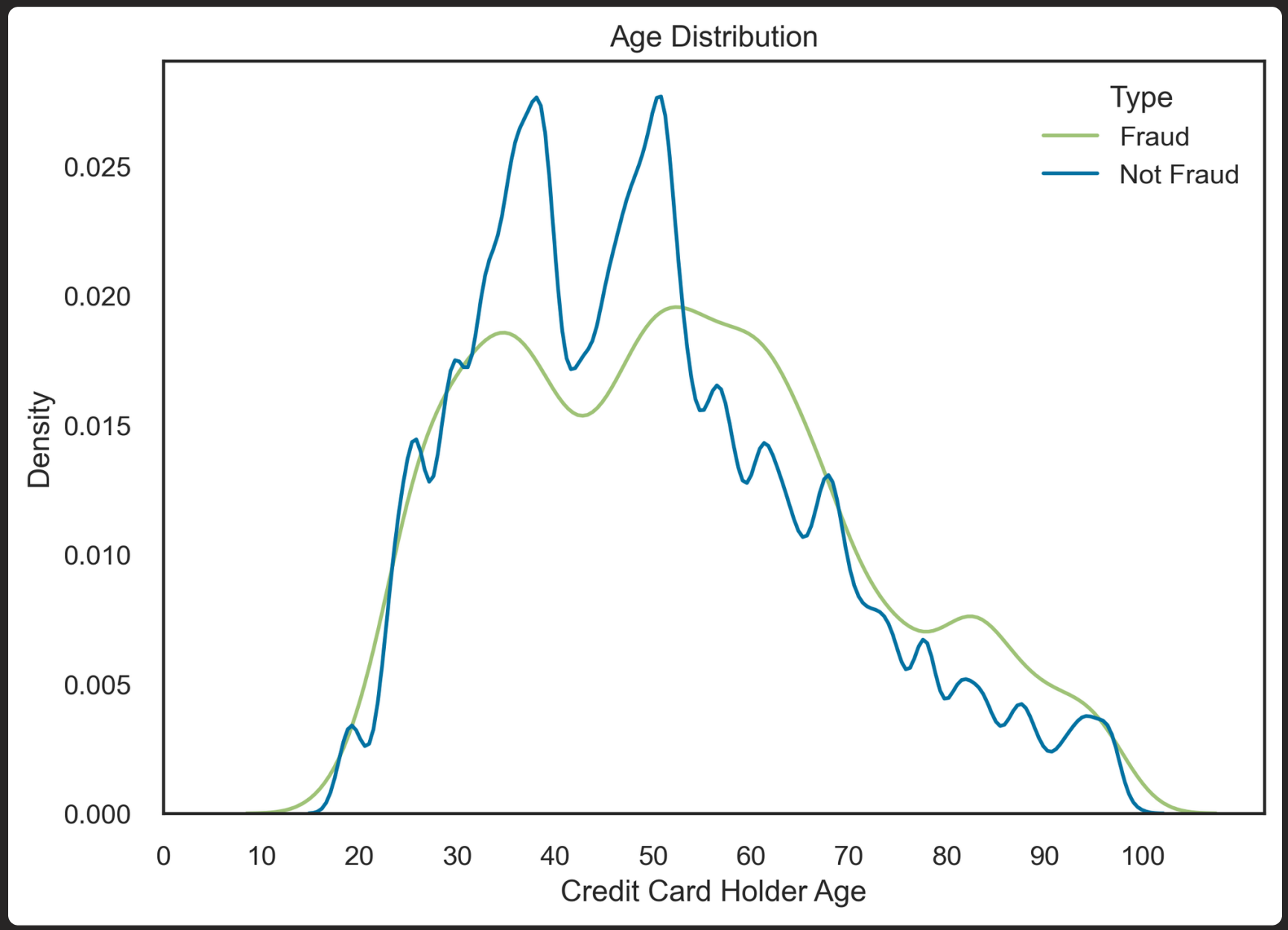
Note that the columns "not_fraud_percentual_vs_total" and "fraud_percentage_vs_total" sums up to 1. That's because we are taking the percentual of each category on total, for fraud and not fraud, and calculating the "fraud_level" metric, that shows which category is more common to have frauds.

As shown above, some categories are more propicious to have frauds than others.

Made with Gamma

Age

Let's see how the Age distribution behaves for fraud and non-fraud transactions.

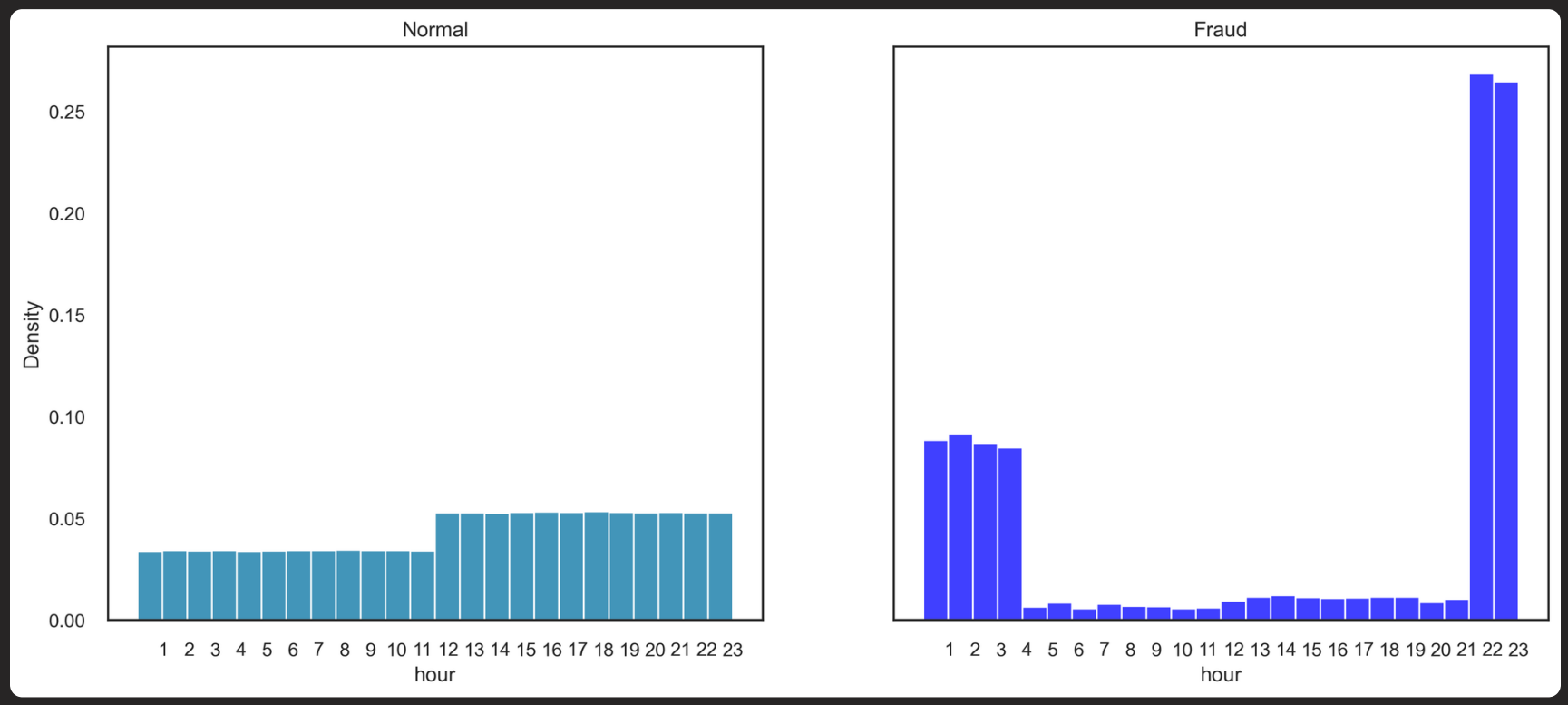


We can observe that in non-fraudulent transactions, there is a peak around 35 years and 40 years, and another one around 50 years. On the other hand, in fraudulent operations, we observe a smoother distribution, with peaks around 35 years and in the range of 50 years to 55 years.

Made with Gamma

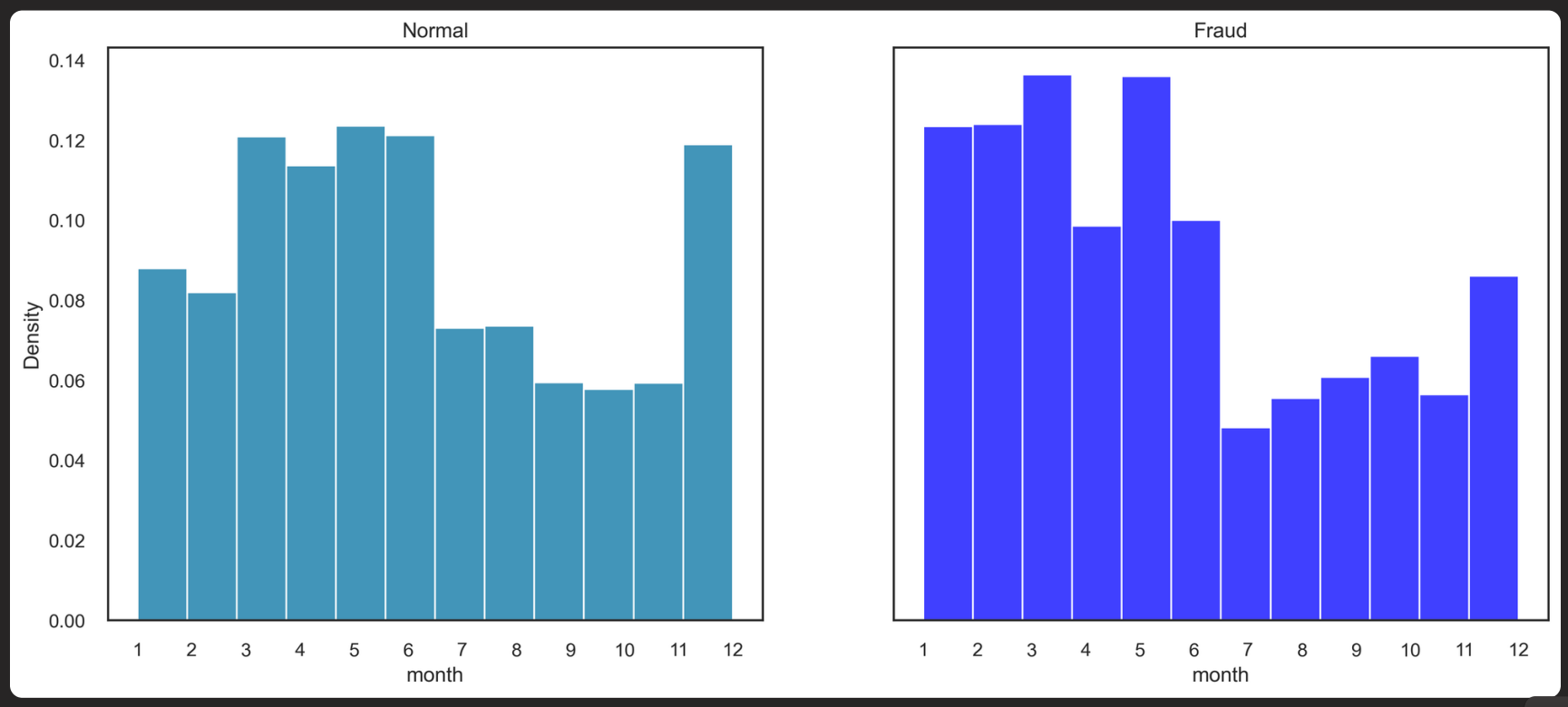
Time

Let's analyze if the time in a day has any impact.



As we can see, there is a clear pattern when it comes to hour in the day. Fraudulent payments happens more frequently around midnight than in normal transactions.

Let's do the same for months!



Made with Gamma

Why is 99% accuracy not good?

As we saw above, this problem has a very imbalanced target class. We have only 0.6% of event occurrence. That means if we do a blind guess, by telling that every transaction is not fraud, we will still get a 99.4% accuracy!

That happens because we will still be getting right all the transactions that are not fraud, which stands for 99.4% of the dataset.

For that reason, it is very dangerous to look at only accuracy in a model. We have plenty of others good metrics to look at, besides accuracy.

Some of them are: Precision, Recall, F1-score, ROC Curves and AUC!

Made with Gamma

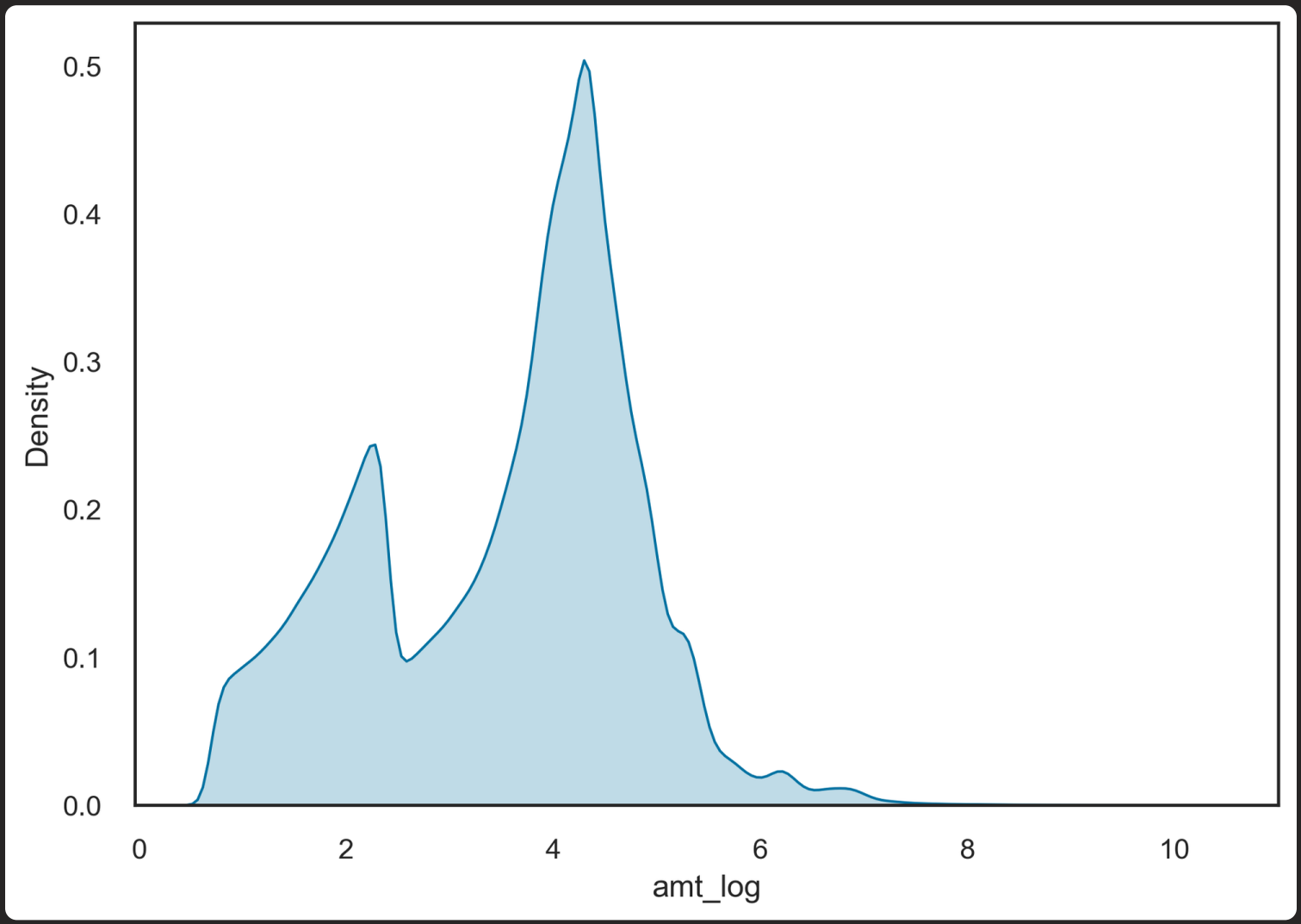
Feature Engineering

First, let's drop some columns that has duplicated or not useful information.

["merchant", "first", "last", "street", "unix_time", "trans_num"]

We saw that the "amt" category has very little spacing between small numbers and large spacing between high numbers. Because of that, we are going to use non logarithm scaling, that increases the distance between small values and reduces de spacing between large ones.

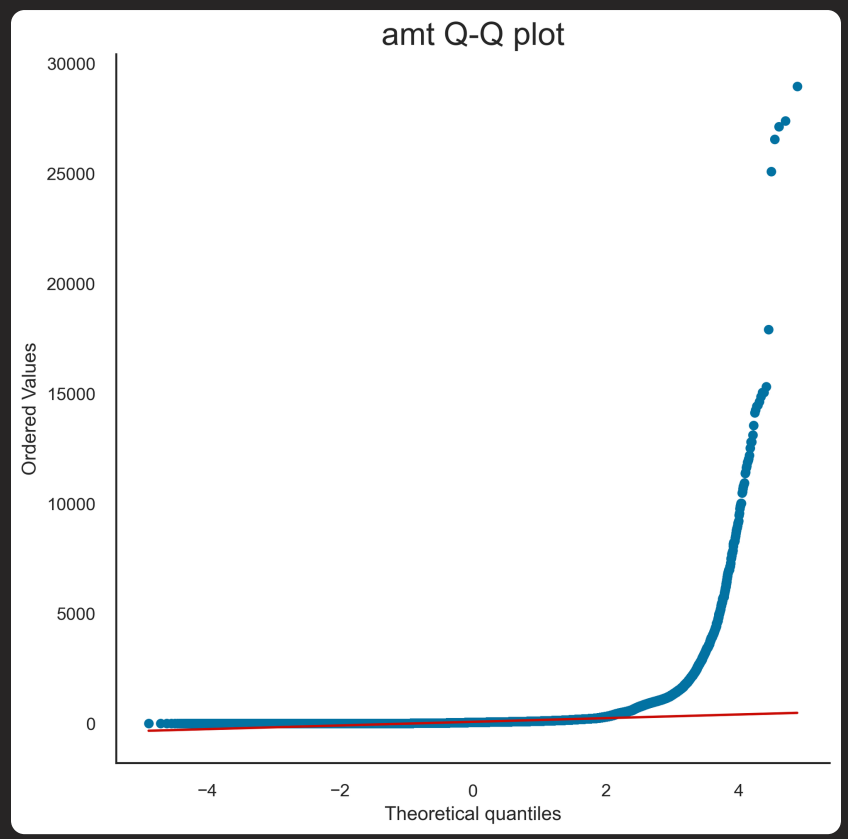
Scaling

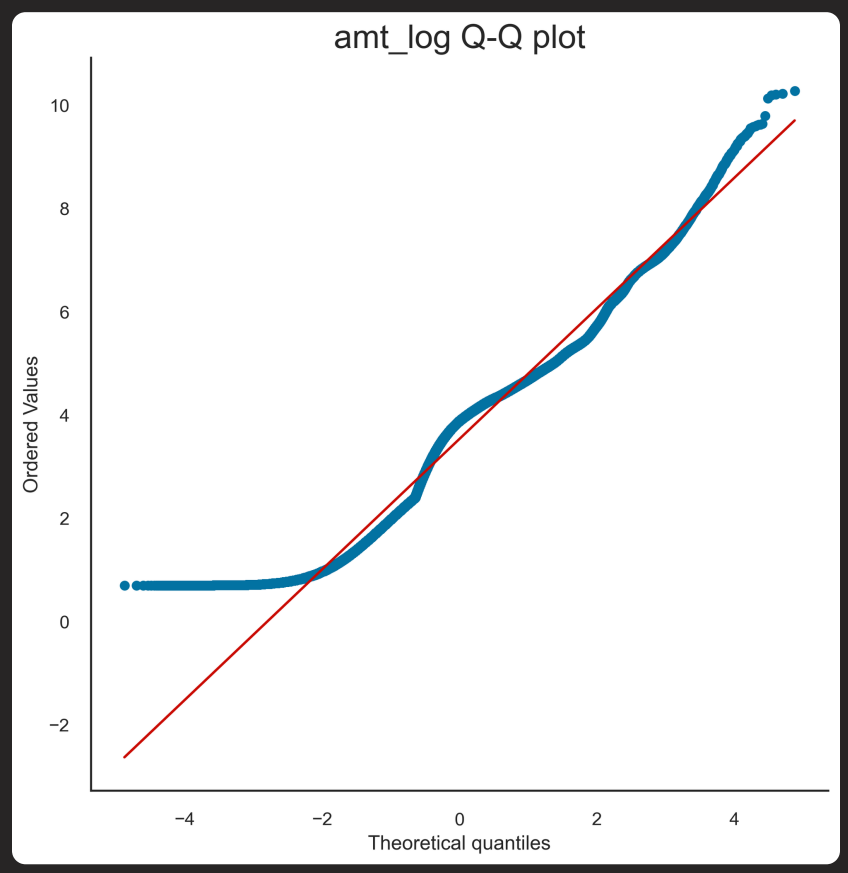


Made with Gamma

"amt"

Let's check the normality of the "amt" feature compared to its transformation "amt_log".





We can see that with the logarithm scaling, the skewness has improved a lot.

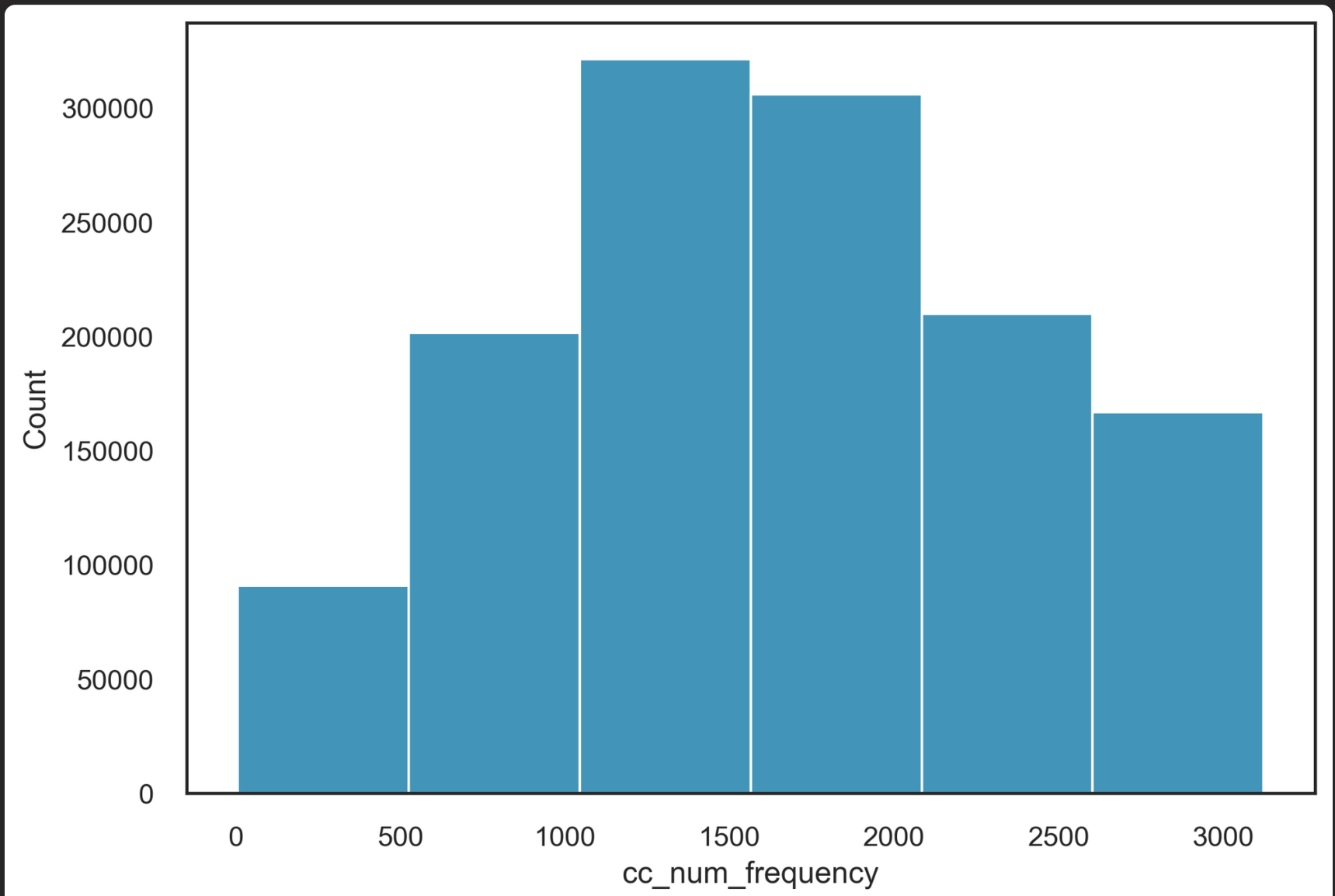
Made with Gamma

Categorical Encoding

Since we have a binary target and categorical variables with high cardinality, we are going to use the WOE (Weight of Evidence) encoder for the features: category, state, city and job.

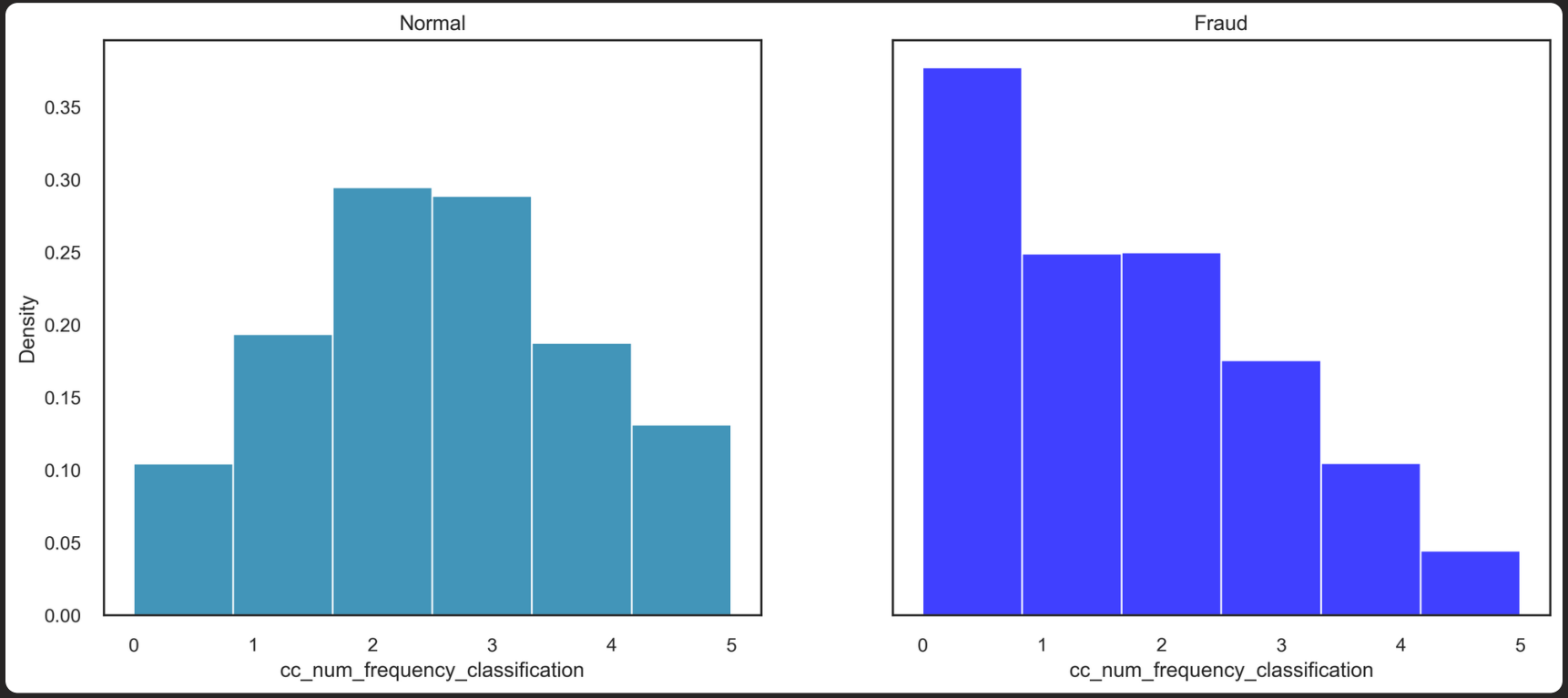
As for gender category, we can simply encode it by using dummies.

We can encode the cc_num (credit card number) variable as well, by counting their frequencies in the database and them dividing them into classes, so we can classify the cards that are used a lot, and the ones that are not used often in the database.



Made with Gamma

Let's see if the distributions are different among the frequency classes for fraudulent and non-fraudulent transactions.



As we can see, there is a clear pattern happening. Frauds are more propitious to happen in credit cards with less use (new ones), and when it comes to normal transactions, it follows a normal distribution.

Made with Gamma

Feature Importance

First, we are going to generate two random vectors to help us compare the feature importances (If a feature has less importance than a random vector, we can suspect that the feature is not good for the model).

- Finally, this is the features that are entering the model.

1. `merch_lat`: Latitude of the merchant location (float64)
2. `age`: Age of the credit card holder (int64)
3. `hour`: Hour of the day when the transaction occurred (int64)
4. `amt_log`: Log-transformed transaction amount (float64)
5. `category_WOE`: Weight of Evidence encoding for the 'category' feature (float64)
6. `city_WOE`: Weight of Evidence encoding for the 'city' feature (float64)
7. `job_WOE`: Weight of Evidence encoding for the 'job' feature (float64)
8. `cc_num_frequency`: Frequency/count of credit card numbers (int64)

These features will serve as the input variables for the model

Made with Gamma

Sampling

performing sampling after splitting data into training and testing sets, must apply the sampling techniques only to the training set. This is to ensure that the testing set remains representative of the original distribution.

SMOTE (Synthetic Minority Over-sampling Technique):

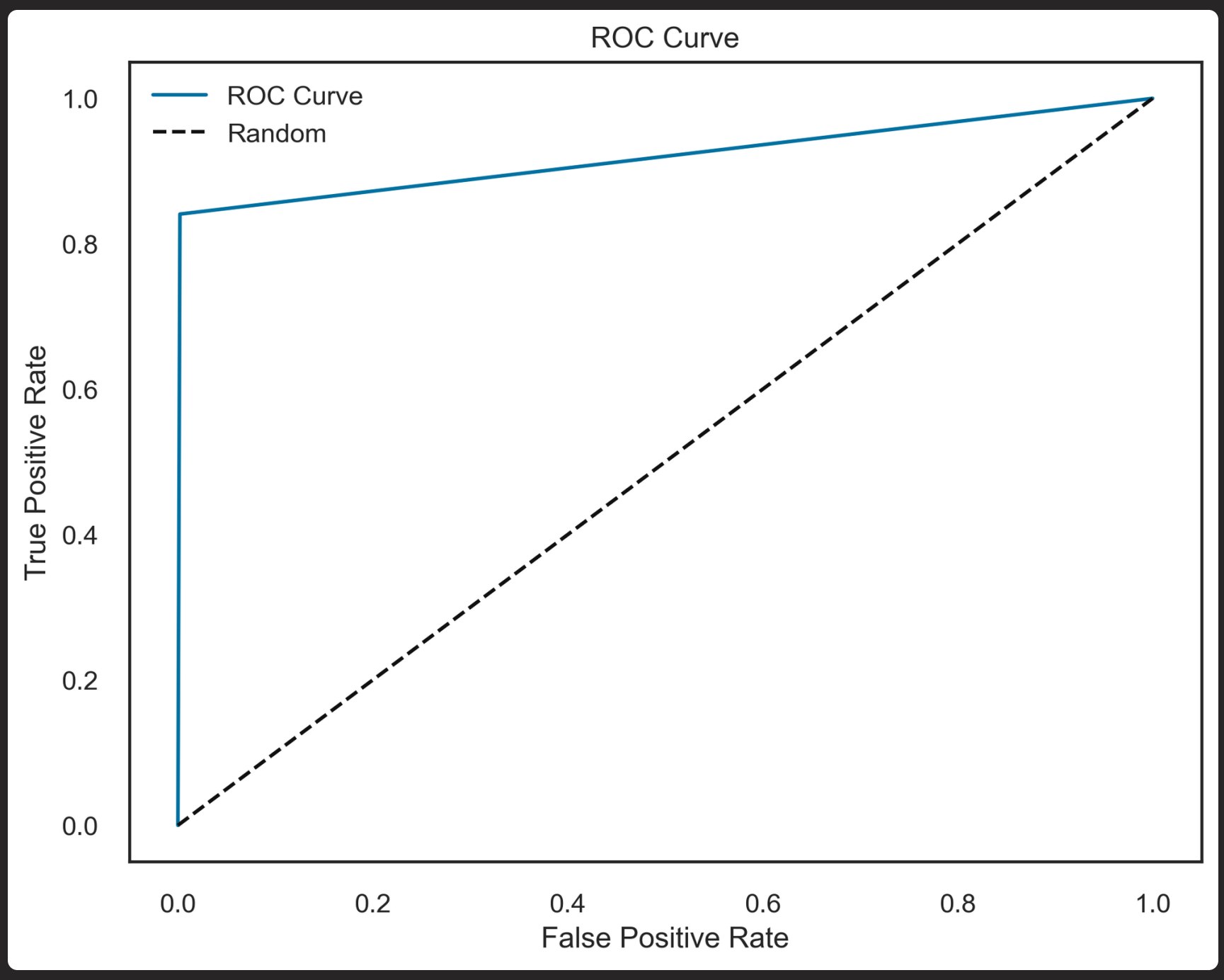
- **Oversampling:** Generates synthetic samples for the minority class based on nearest neighbors.
- **Pros:** Addresses overfitting issues of random oversampling.
- **Cons:** May introduce noise.

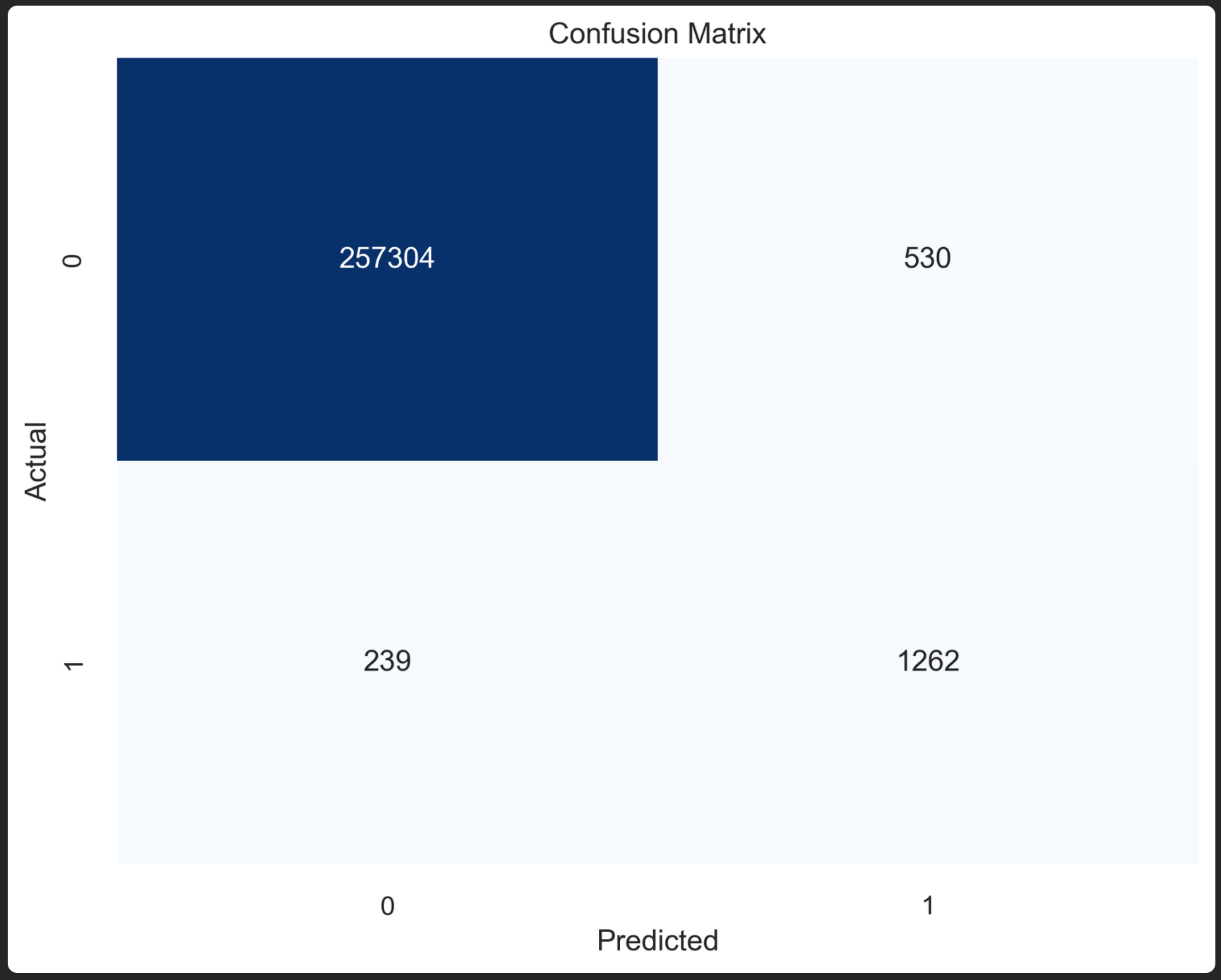
Using SMOTE

```
X_train shape:(1037340, 8)
y_train shape:(1037340,)
X_test shape:(259335, 8)
y_test shape:(259335,)
X_train_resampled shape:(2062670, 8)
y_train_resampled shape:(2062670,)
```

Made with Gamma

DecisionTree





Accuracy: 0.9970347234272273

ROC AUC: 0.9193586159844609

F1 Score: 0.7664743395080472

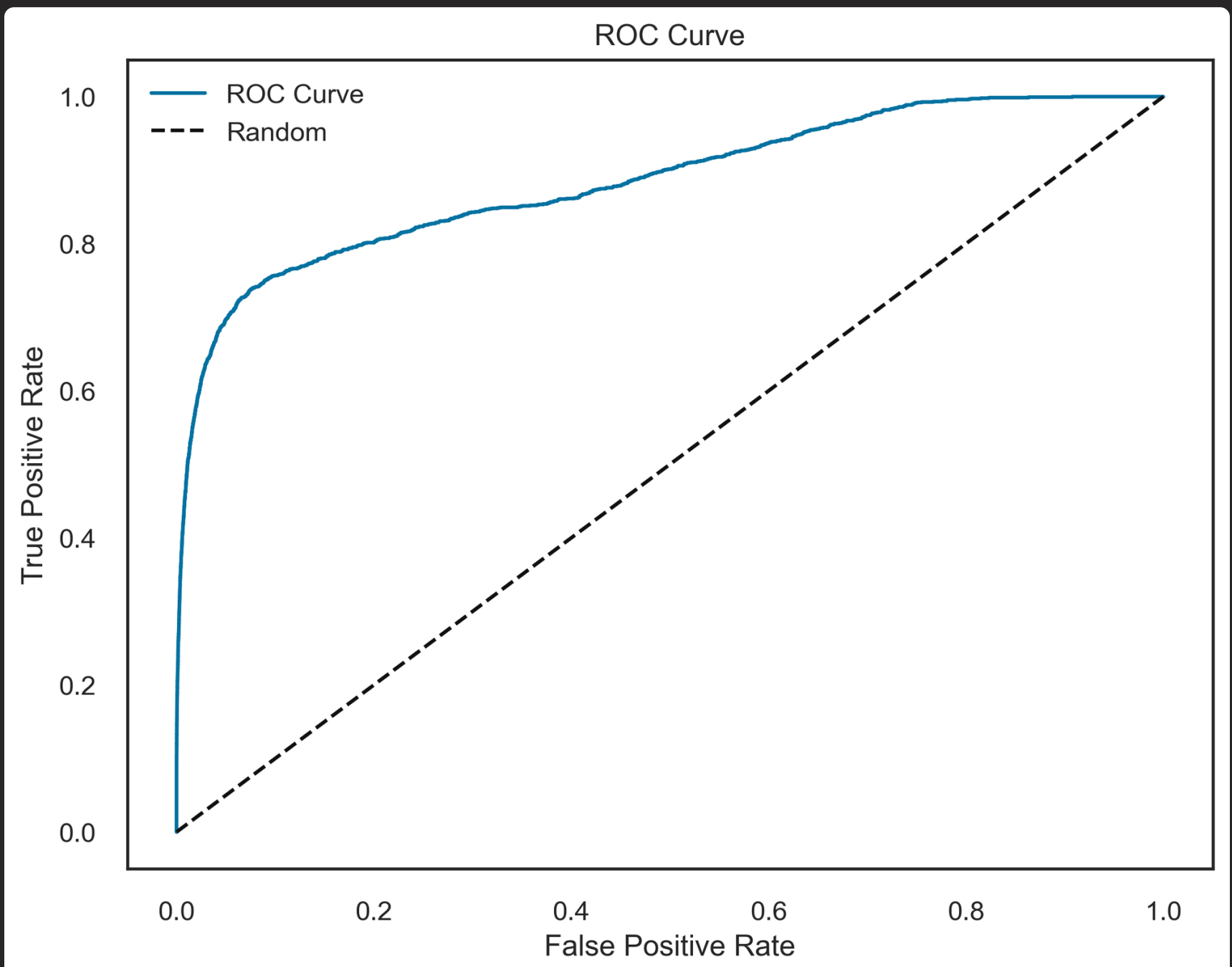
Confusion Matrix:

[[257304 530]

[239 1262]]

Made with Gamma

Logistic Regression:



Accuracy: 0.8291784757167371

ROC AUC: 0.8849003715506917

F1 Score: 0.050944770555722184

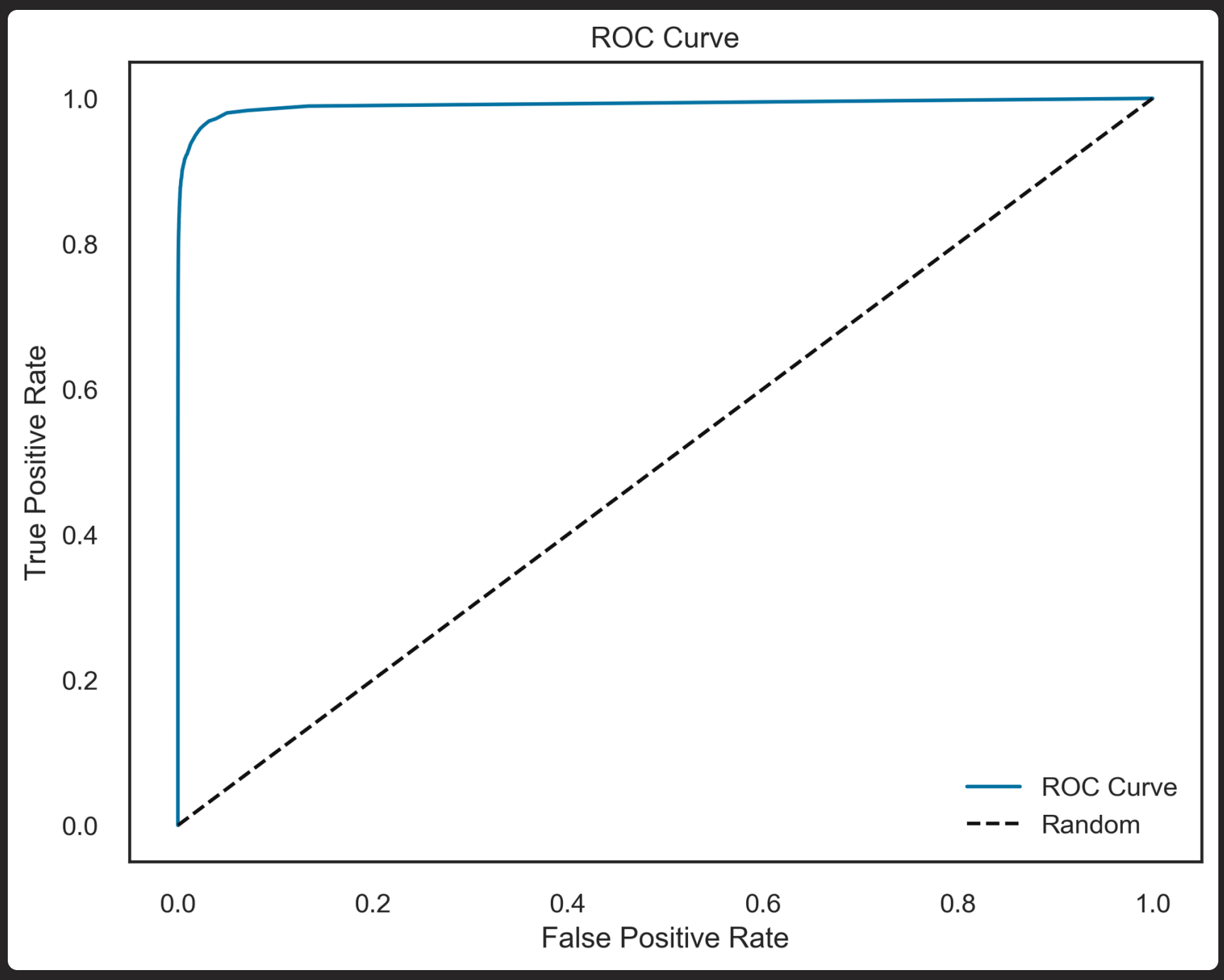
Confusion Matrix:

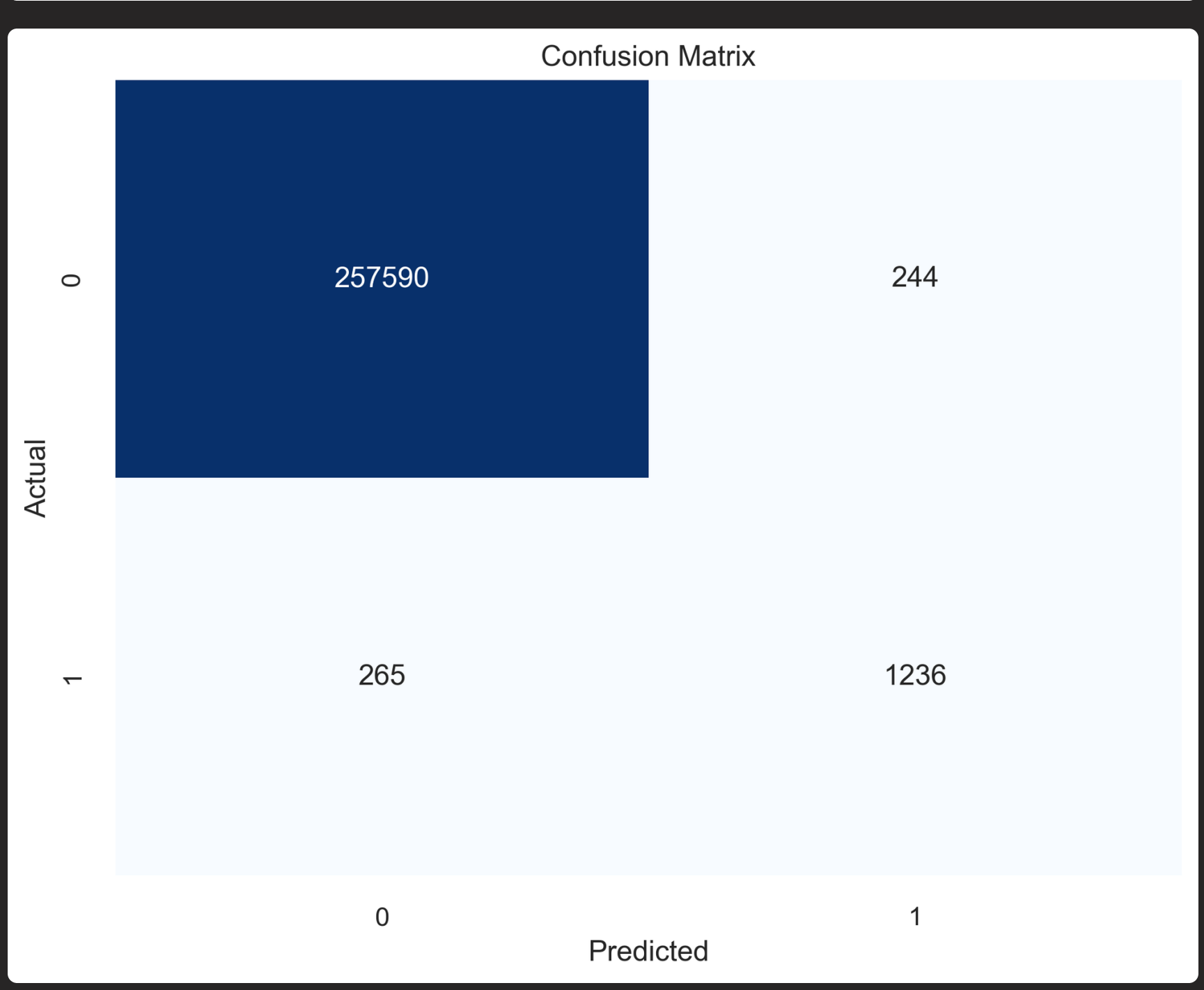
[[213846 43988]

[312 1189]]

Made with Gamma

Random Forest





Accuracy: 0.9980372876780997

ROC AUC: 0.9914719789057839

F1 Score: 0.8292519288829253

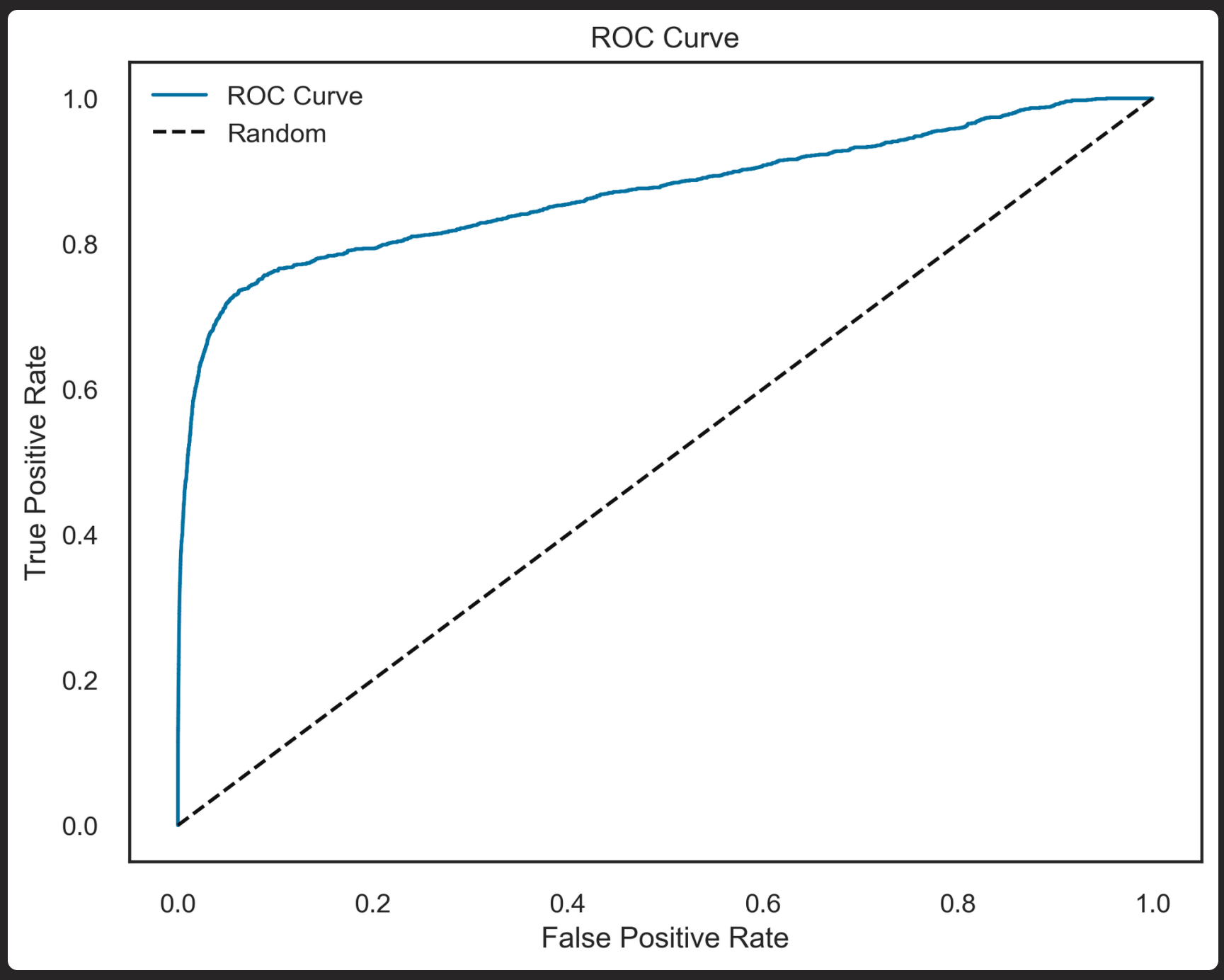
Confusion Matrix:

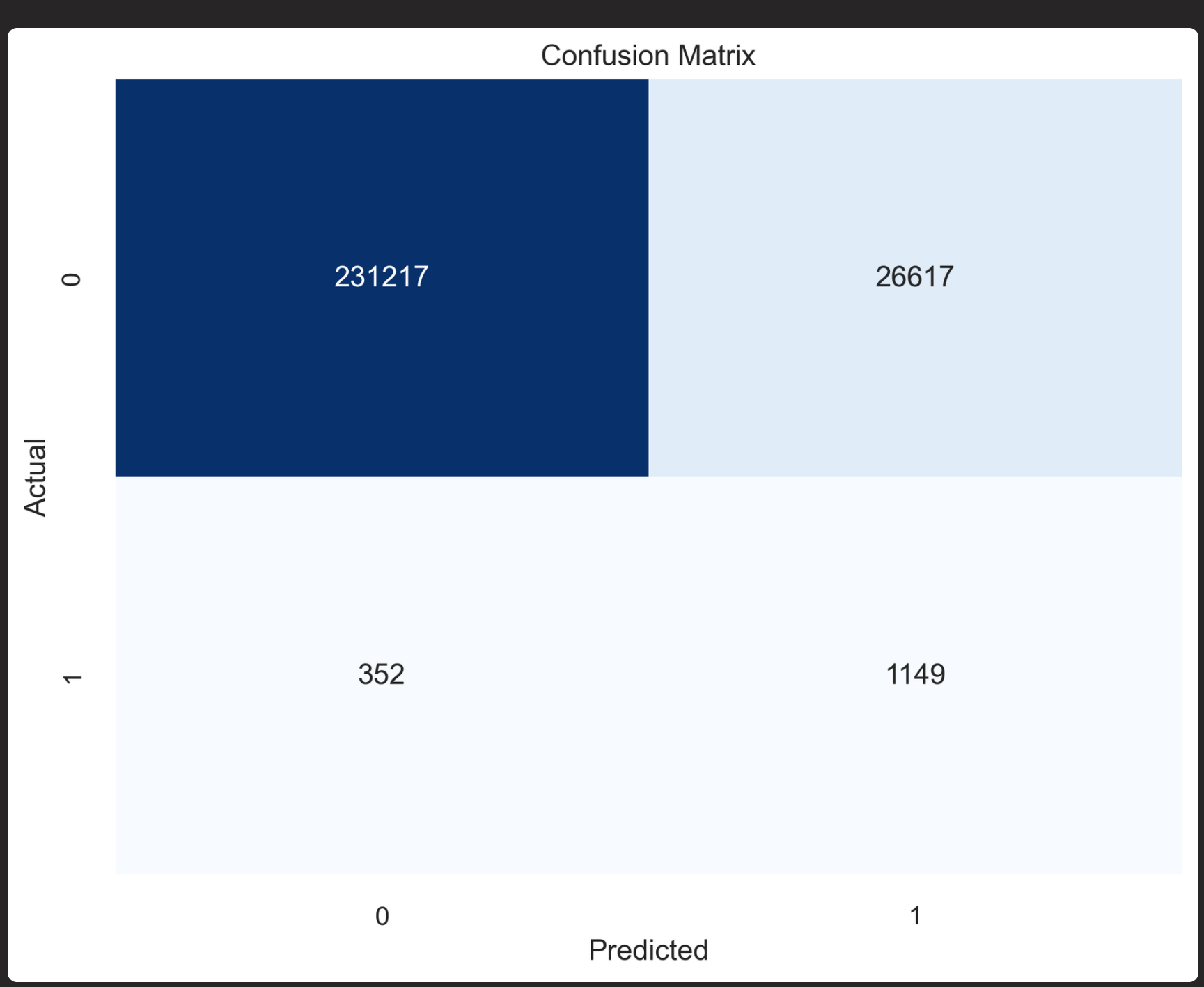
[[257590 244]

[265 1236]]

Made with Gamma

Naive Bayes





Accuracy: 0.8960070950700831

ROC AUC: 0.8704104645838653

F1 Score: 0.07851846789899886

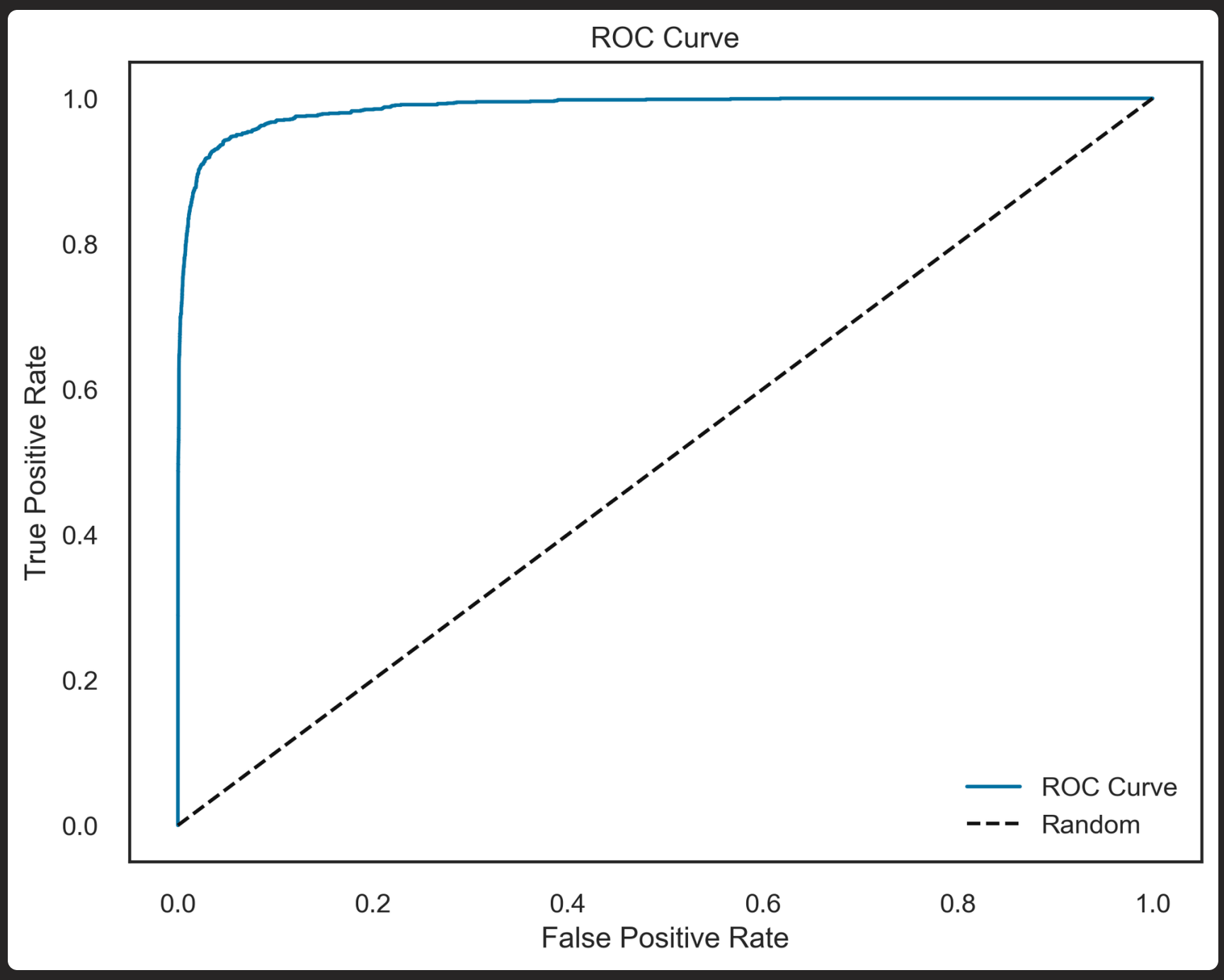
Confusion Matrix:

[[231217 26617]

[352 1149]]

Made with Gamma

gradient_boosting



Accuracy: 0.9800875315711338

ROC AUC: 0.9874962324503425

F1 Score: 0.3409903011740684

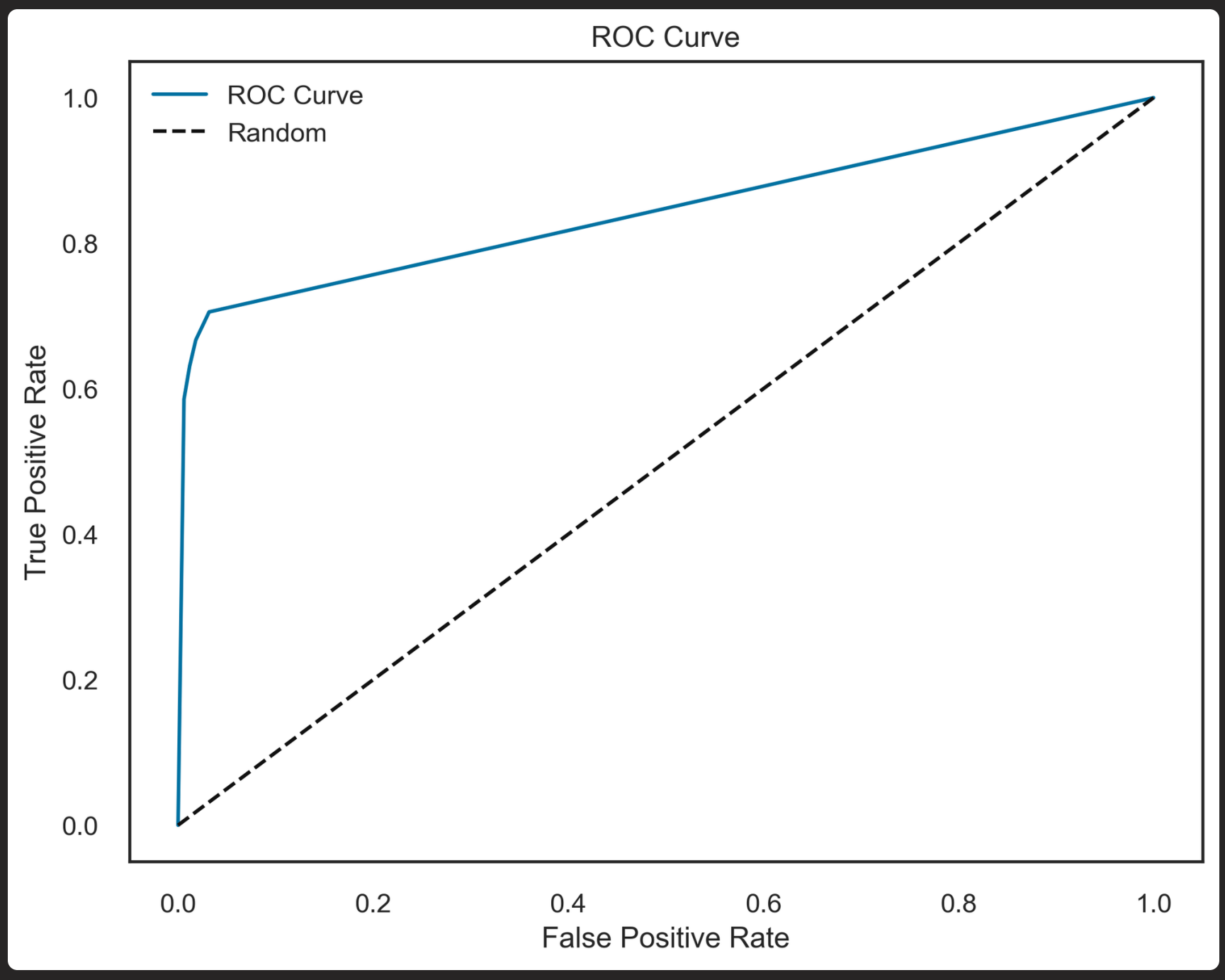
Confusion Matrix:

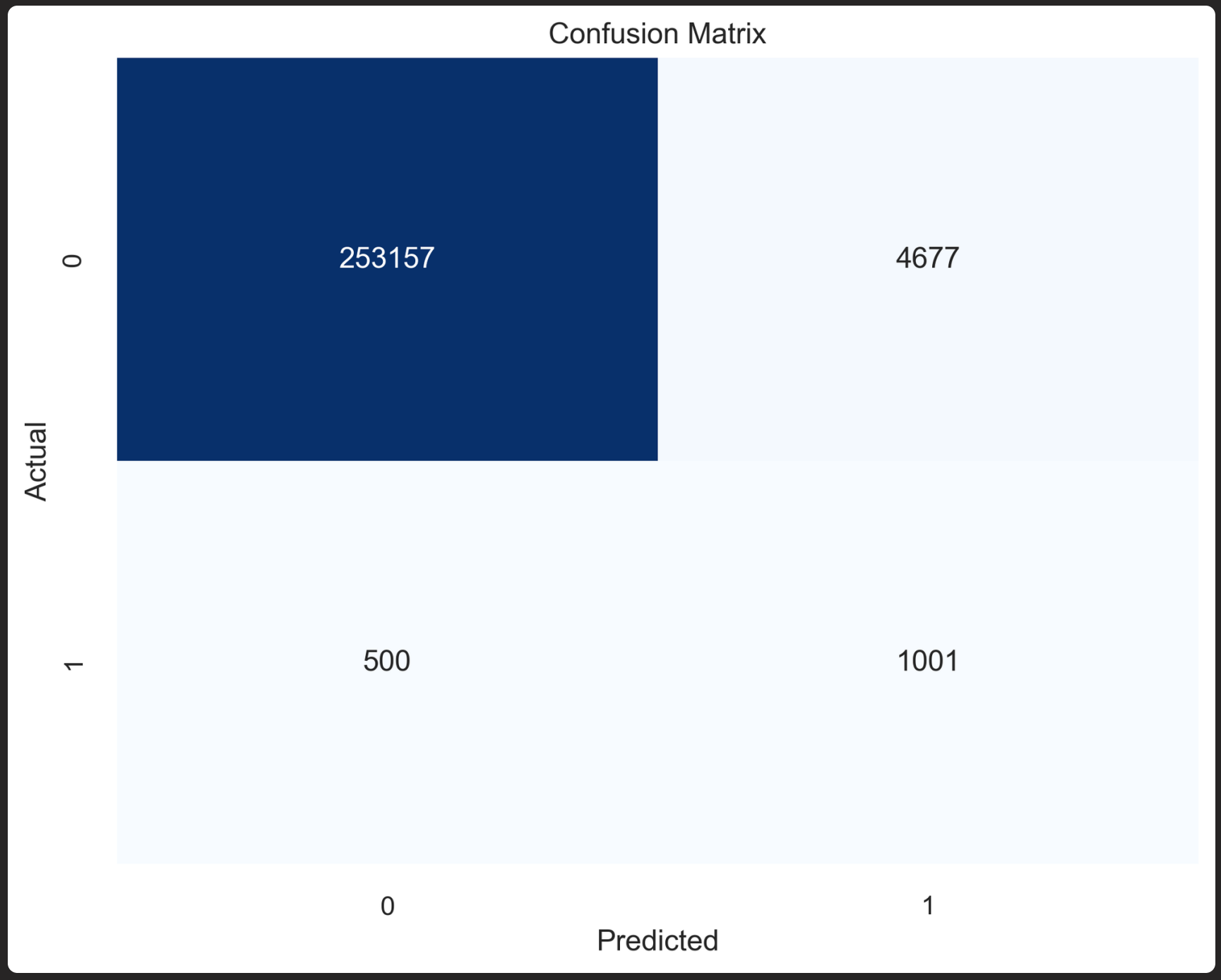
[[252835 4999]

[165 1336]]

Made with Gamma

K-nearest neighbor (KNN)





Accuracy: 0.9800374033585902

ROC AUC: 0.8443800587766428

F1 Score: 0.27886892324836327

Confusion Matrix:

[[253157 4677]

[500 1001]]

Made with Gamma

Support Vector Machine (SVM):

Accuracy: 0.5082769390942218

ROC AUC: 0.6715313234942849

F1 Score: 0.016580422762221315

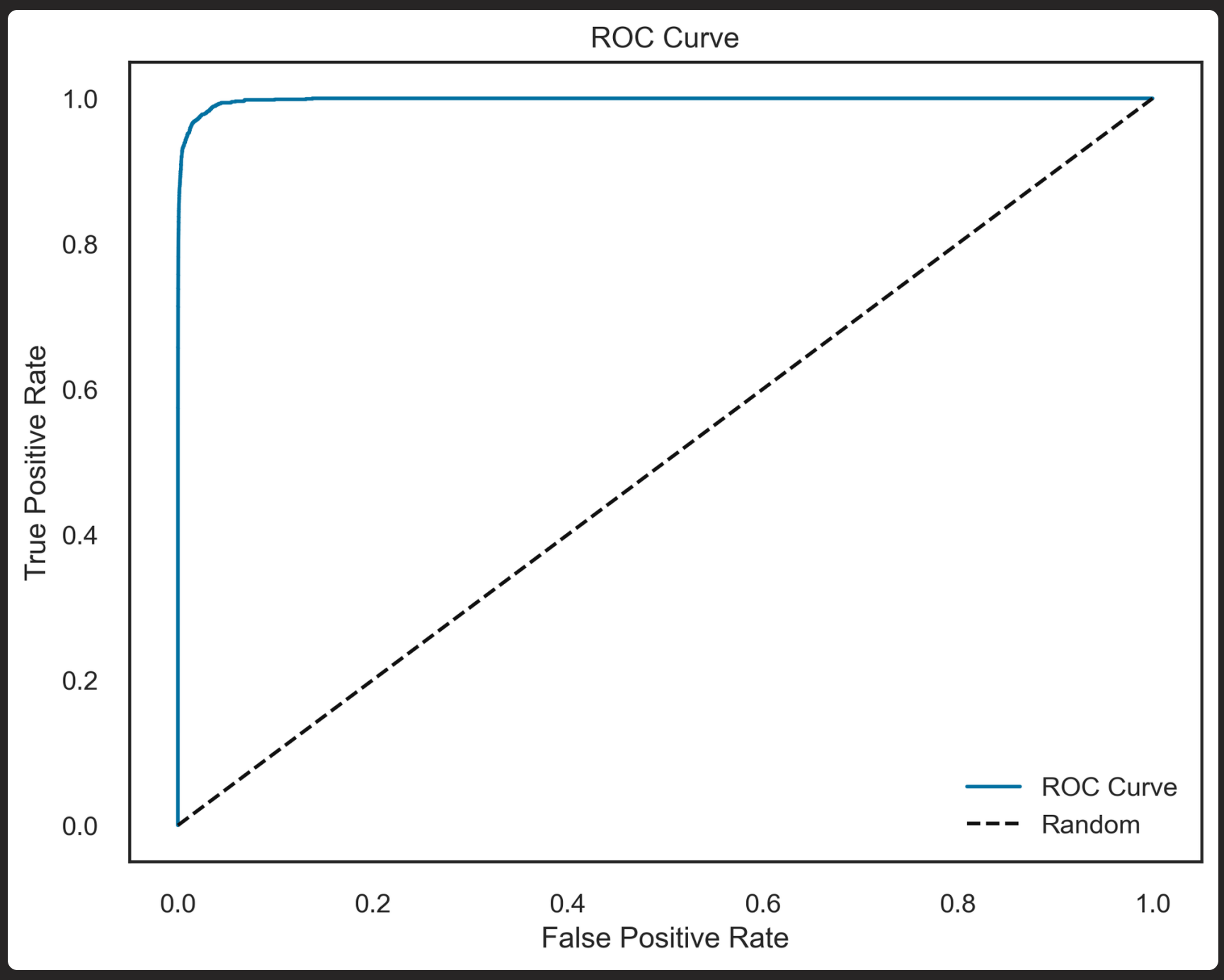
Confusion Matrix:

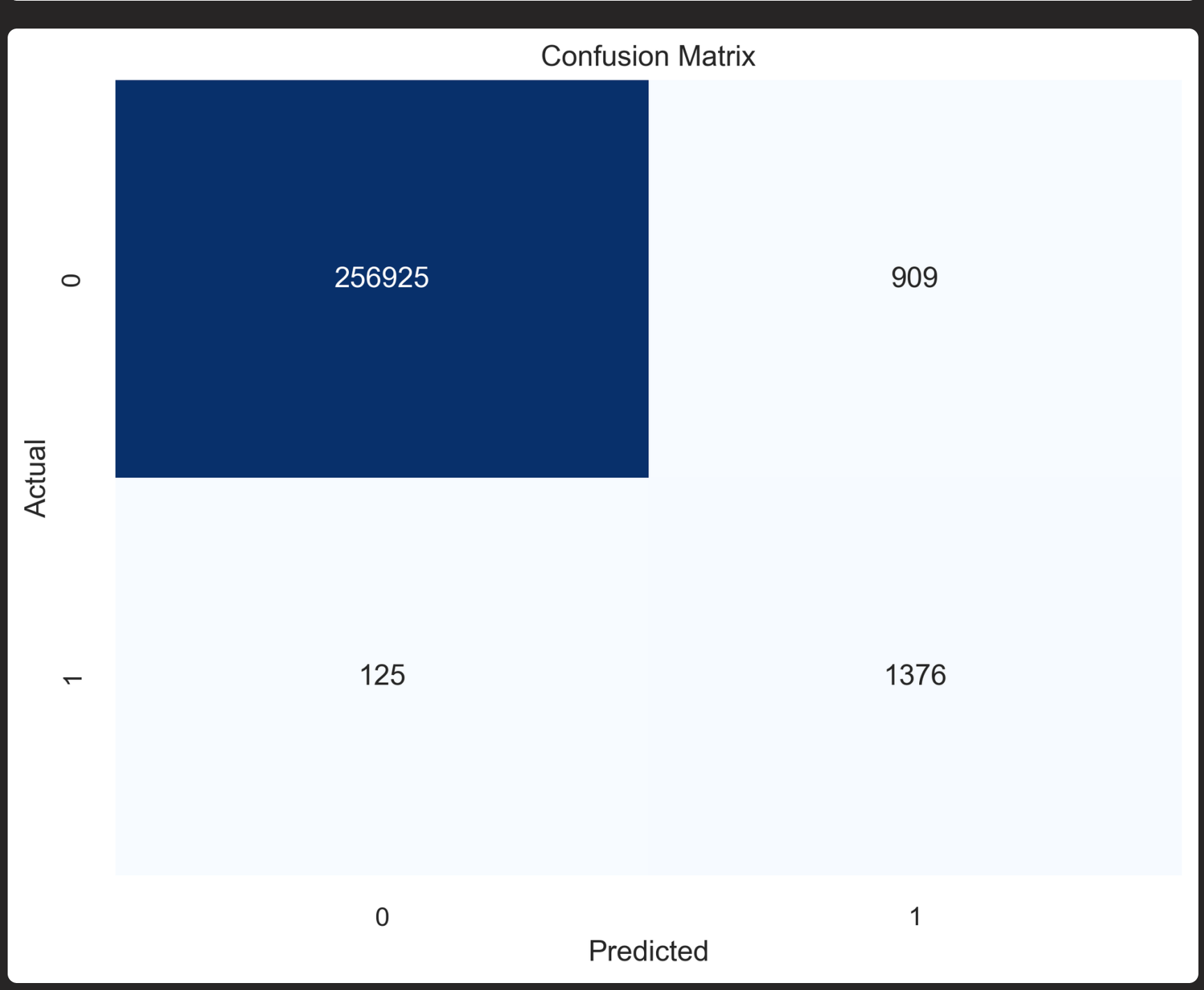
[[130739 127095]

[426 1075]]

Made with Gamma

LightGBM





Accuracy: 0.9960128790946073

ROC AUC: 0.998081612266246

F1 Score: 0.7268885367142102

Confusion Matrix:

[[256925 909]

[125 1376]]

Made with Gamma

Model Evaluation Report:

1. Logistic Regression:

- **Accuracy:** 82.92%
- **ROC AUC:** 88.49%
- **F1 Score:** 5.09%

Comments: The model seems to struggle with precision, potentially due to the convergence warning. It might benefit from further hyperparameter tuning.

2. Random Forest:

- **Accuracy:** 99.80%
- **ROC AUC:** 99.15%
- **F1 Score:** 82.93%

Comments: The Random Forest model exhibits excellent performance with high accuracy, ROC AUC, and F1 Score. It appears robust to the imbalanced nature of the data.

Made with Gamma

3. Naive Bayes (GaussianNB):

- **Accuracy:** 89.60%
- **ROC AUC:** 87.04%
- **F1 Score:** 7.85%

Comments: Naive Bayes shows decent performance, especially in terms of accuracy. However, the F1 Score indicates room for improvement, possibly through more advanced models.

4. Gradient Boosting:

- **Accuracy:** 98.01%
- **ROC AUC:** 98.75%
- **F1 Score:** 34.10%

Comments: Gradient Boosting performs well, showcasing high accuracy and ROC AUC. However, the F1 Score suggests a trade-off between precision and recall.

5. K-nearest neighbor (KNN):

- **Accuracy:** 98.00%
- **ROC AUC:** 84.44%
- **F1 Score:** 27.89%

Comments: KNN provides good accuracy but demonstrates challenges in achieving a balance between precision and recall, as indicated by the F1 Score.

Made with Gamma

6. Support Vector Machine (SVM):

- **Accuracy:** 50.83%
- **ROC AUC:** 67.15%
- **F1 Score:** 1.66%

Comments: SVM exhibits relatively lower performance compared to other models, especially in terms of ROC AUC and F1 Score. The updated results suggest a significant decrease in performance.

7. LightGBM:

- **Accuracy:** 99.60%
- **ROC AUC:** 99.81%
- **F1 Score:** 72.69%

Comments: LightGBM performs exceptionally well, demonstrating high accuracy, ROC AUC, and F1 Score. It appears robust to the imbalanced nature of the data.

Made with Gamma

Overall Comments:

- **Random Forest** and **LightGBM** stand out as the top-performing models across all metrics.
- **Naive Bayes** and **SVM** show decent but comparatively lower performance, especially in terms of precision.
- **Further tuning and experimentation** may enhance the performance of individual models.

Made with Gamma

Grid Search

as Random Forest and LightGBMmodel have a great result. performed **Hyperparameter Tuning.**

Best Parameters for LightGBM: {'num_leaves': 50, 'n_estimators': 100, 'min_child_samples': 20, 'learning_rate': 0.1, 'boosting_type': 'dart'}
Best F1 Score for LightGBM: 0.9984901854459345

Best Parameters for Random Forest: {'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_depth': 30, 'bootstrap': False}
Best F1 Score for Random Forest: 1.0

Made with Gamma

Conclusion

In this problem, it is crucial to look at other metrics besides accuracy. As we saw, accuracy is one of the worst metrics for this problem, because of the natural imbalance of the dataset.

Since our event is related to fraud transactions, i assume the worst cenarios was getting high false negatives (transactions that we identify as non-frauds and ended up being fraudulents). Hence, we used an aproach to consider the recall metric being the most important for the model.

Also, the precision of the model is very sensitive, because of the target imbalance. As you can see, only 0.71% or normal transactions were false positives, and it dropped the precision to 33%. That is because a small percentage of normal transactions if predicted wrong, means a high number of events compared to the fraud events in dataset.

The best approach to this problem would be having a way to calculate the mean cost of each false positive and false negative. This way, we could approach this tradeoff with better understanding of how much false positives and false negatives costs for the company.

Made with Gamma