ICS 474 Project Report

Term 241

Instructor name: Dr. Muzammil Behzad

ID	Name
Mamdouh Almatrafi	202043260
Hamza Alnour Abdallah	202044820
Ahmed Alsayed	202045040

Part 1: Data Understanding and Exploration

1. Dataset Overview:

The chosen dataset is **credit card Transactions**, and the goal of this dataset is to collect a large number of detailed records related to credit card purchases such as transaction time, payment amounts, merchant information, and related customer details.

As a result, People can use this dataset to apply their analyses on various aspects, such as Fraud Detection, Customer Segmentation, and Transaction Classification and so on.

Hugging Face was the source that dataset was coming from. The problem domain it addresses includes financial, customer insights, and spending patterns.

2. Feature Description:

Feature name	Data Type	Category	Description
Unnamed: 0	int64	Numerical	Not shown on the dataset.
trans_date_trans_tim	object	Categorical	calculate timestamp of the transaction.
cc_num	int64	Numerical	Credit card number
merchant	object	Categorical	Merchant or store where the transaction occurred.
category	object	Categorical	Type of transaction (e.g., grocery, entertainment).
amt	float64	Numerical	calculate the amount of the transaction.
first	object	Categorical	First name of the cardholder.
last	object	Categorical	last name of the cardholder.

gender	**object**	**Categorical**	**Gender of the cardholder.**
street	**object**	**Categorical**	**Street address of the cardholder.**
city	**object**	**Categorical**	**City where the cardholder resides.**
state	**object**	**Categorical**	**state where the cardholder resides.**
zip	**int64**	**Numerical**	**Zip code of the cardholder.**
lat	**float64**	**Numerical**	**Geographical coordinates of the transaction.**
long	**float64**	**Numerical**	**Geographical coordinates of the transaction.**
city_pop	**int64**	**Numerical**	**Population of the city where the transaction occurred.**
job	**object**	**Categorical**	**Occupation of the cardholder.**
dob	**object**	**Categorical**	**Date of birth of the cardholder.**
trans_num	**object**	**Categorical**	**Unique transaction number.**
unix_time	**int64**	**Numerical**	**Unix timestamp of the transaction.**
merch_lat	**float64**	**Numerical**	**Geographical coordinates of the merchant.**

merch_long	**float64**	**Numerical**	**Geographical coordinates of the merchant.**
is_fraud	**int64**	**Numerical**	**Binary indicator of whether the transaction is fraudulent.**
merch_zipcode	**float64**	**Numerical**	**Geographical coordinates of the merchant.**

The target variable is is_fraud since it is used for fraud detection analysis.

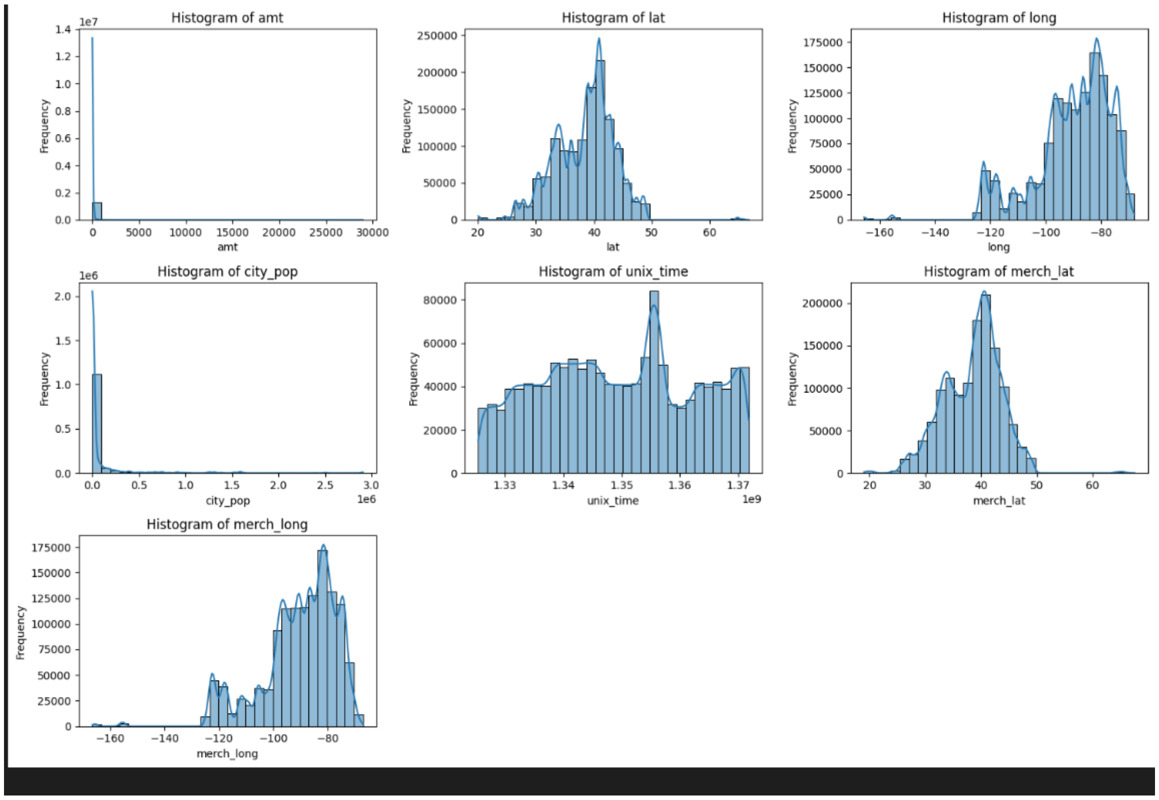
3. **Dataset Structure:**
   The dataset contains 1296675 rows, 24 columns and 31120200 elements.

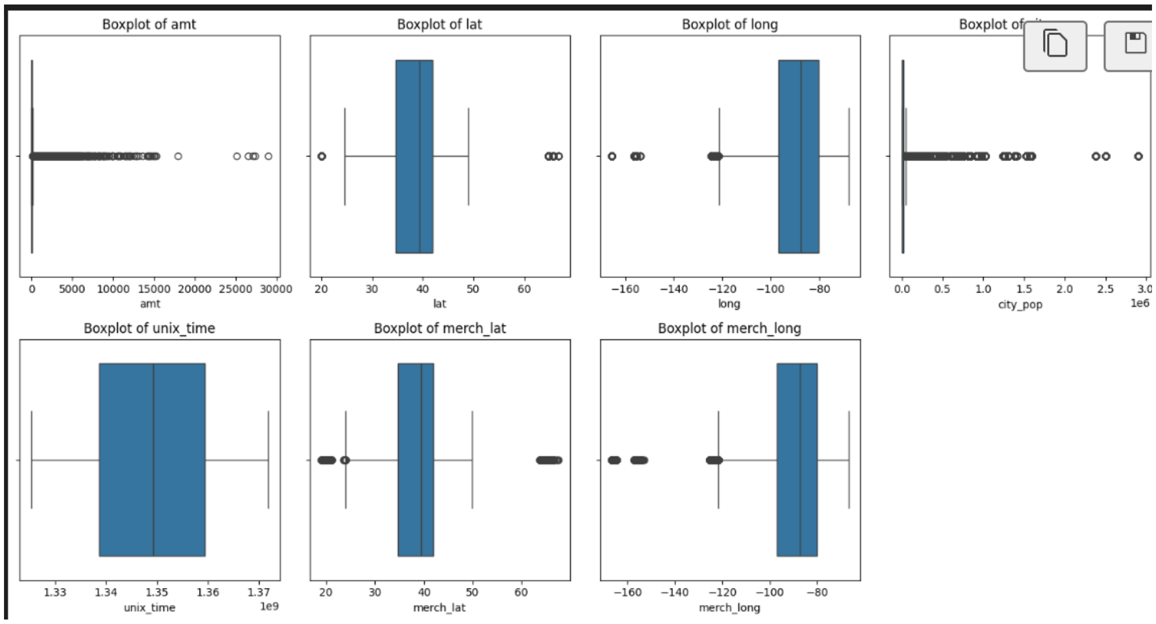
4. **Missing Values and Duplicates:**
   All the columns don't have missing value except merch_zipcode contain 195973 missing values. There are no duplicate rows and columns. As a result, the dataset is almost clean, but the merch_zipcode column affects our analysis when merchant's location is important. Due to no duplicate on both rows and columns, there's no risk of redundant data affecting the analysis and decrease computation time performing on the module and analysis.
5. **Statistical Summary:**
   The amt (transaction amount) column shows a wide range with most transactions around 39.2, but some transactions have exceeded 27,500, suggesting a few very high-value purchases. The city_pop column varies largely, with most cities having populations around 3,124, but some cities have reached over 3 million, which could impact

location-based analysis. Fraud (is_fraud column) is quite rare because it shows that only about 0.57% of transactions are fraudulent. The average transaction amount is approximately $64, but the values range widely, with some reaching nearly $28,500. The timestamps are captured in Unix time, with transactions spanning a range close to 5 years. Zip codes associated with transactions show a median around 49,247, indicating that transactions are mostly centered in typical U.S. regions. Latitude and longitude data suggest that most activities occur within the continental U.S., with customer and merchant locations showing similar geographic distributions.

6. Data Distribution:

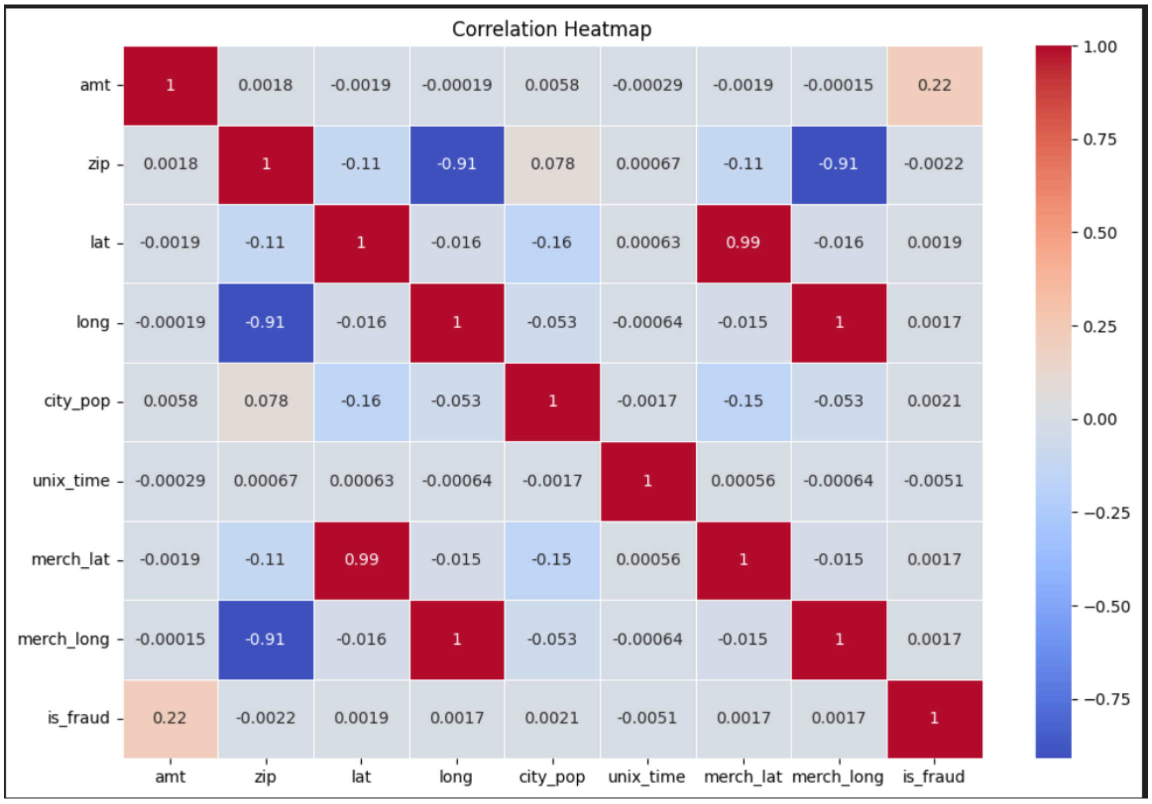
We use Histogram and Boxplot on each numerical feature:





7. Correlation Analysis:

The target variable is fraud. From the heatmap graph: amt column shows a strong correlation with fraud (0.18), so this suggests that higher transaction amounts might be related to fraudulent activities. The three-column zip code, latitude, and longitude show very weak correlation close to zero (around -0.002 to 0.001). As a result, the location data of customers and merchants does not have a strong relationship with whether a transaction is fraudulent or not. The two-column city_pop and unix_time show very minimal correlation with fraud (around 0.0031 and -0.0007 respectively) showing that the size of a city or the time of a transaction doesn't affect whether a transaction is fraudulent. The two-column merch_lat and merch_long show very weak correlation close to zero (-0.0015 to 0.0018), so the merchant’s location doesn’t affect fraud.



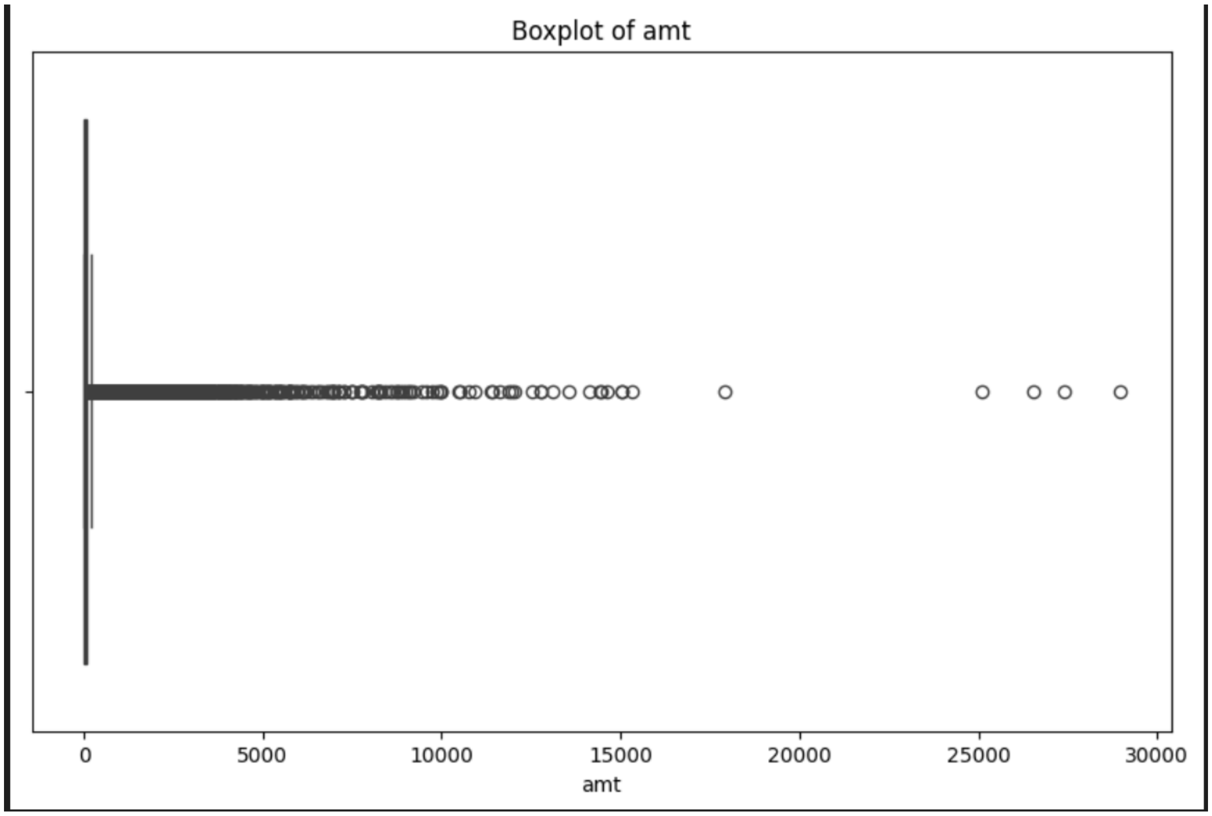
	amt	zip	lat	long	city_pop	unix_time	merch_lat	merch_long	is_fraud
amt	1	0.0018	-0.0019	-0.00019	0.0058	-0.00029	-0.0019	-0.00015	0.22
zip	0.0018	1	-0.11	-0.91	0.078	0.00067	-0.11	-0.91	-0.0022
lat	-0.0019	-0.11	1	-0.016	-0.16	0.00063	0.99	-0.016	0.0019
long	-0.00019	-0.91	-0.016	1	-0.053	-0.00064	-0.015	1	0.0017
city_pop	0.0058	0.078	-0.16	-0.053	1	-0.0017	-0.15	-0.053	0.0021
unix_time	-0.00029	0.00067	0.00063	-0.00064	-0.0017	1	0.00056	-0.00064	-0.0051
merch_lat	-0.0019	-0.11	0.99	-0.015	-0.15	0.00056	1	-0.015	0.0017
merch_long	-0.00015	-0.91	-0.016	1	-0.053	-0.00064	-0.015	1	0.0017
is_fraud	0.22	-0.0022	0.0019	0.0017	0.0021	-0.0051	0.0017	0.0017	1

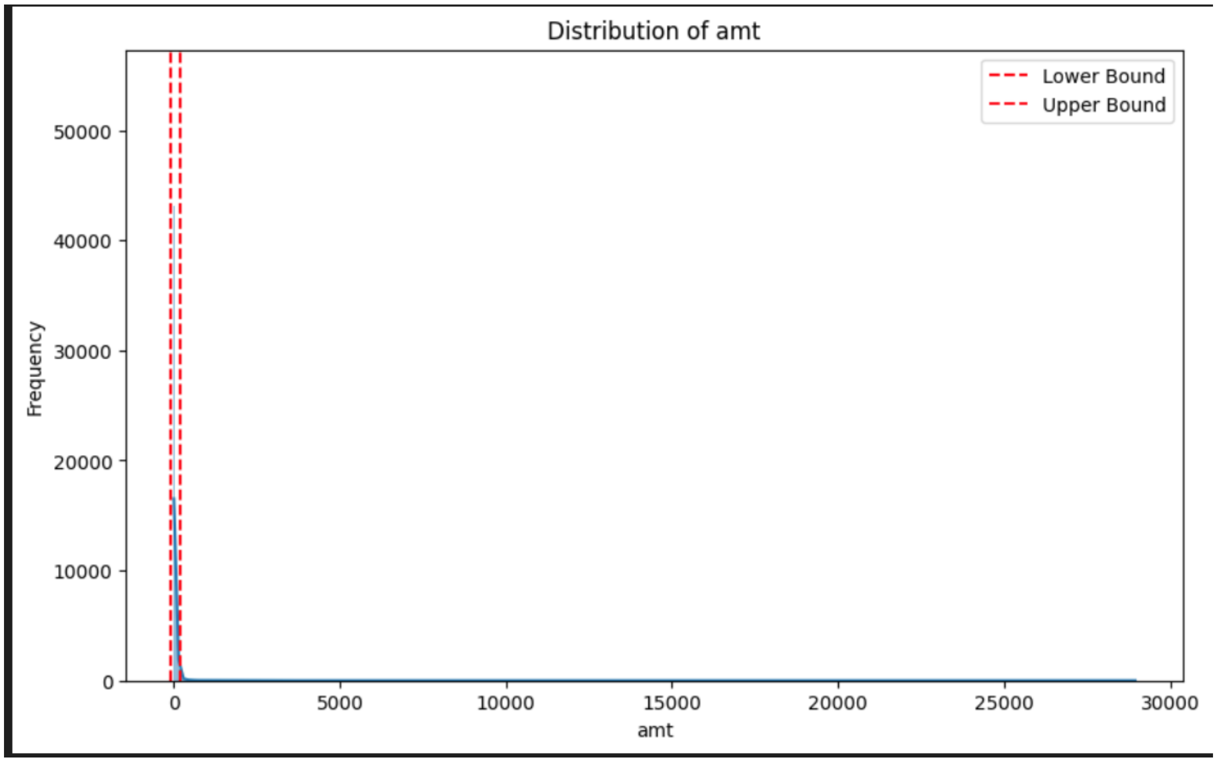
8. Outlier Detection:

We analyzed the amt column (most related column to fraud) to search for outliers. From the boxplot, most transaction amounts are between 0 and 10,000. After 15,000, there is one outlier between 15,000 and 20,000. Additionally, there are four outliers with values between 25,000 and 30,000.

In histogram, most frequency values are located around zero and the rest amt values at zero frequency. Because the graph is right skewed, the outlier on the right side.

As a result, these high amounts could indicate potential fraud, and it affects our detection of fraud.

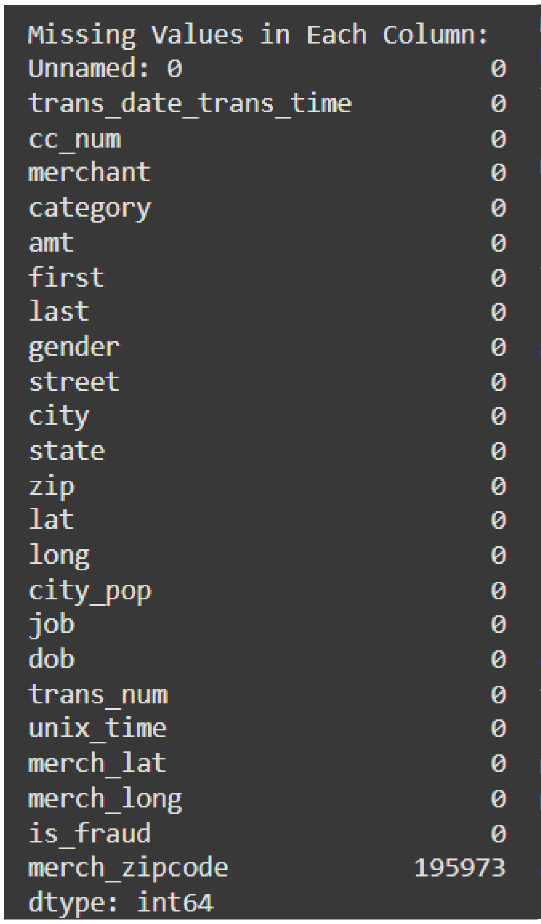

Boxplot of amt
0
5000
10000
15000
20000
25000
30000
amt


Distribution of amt
Lower Bound
Upper Bound
Frequency
50000
40000
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000
amt

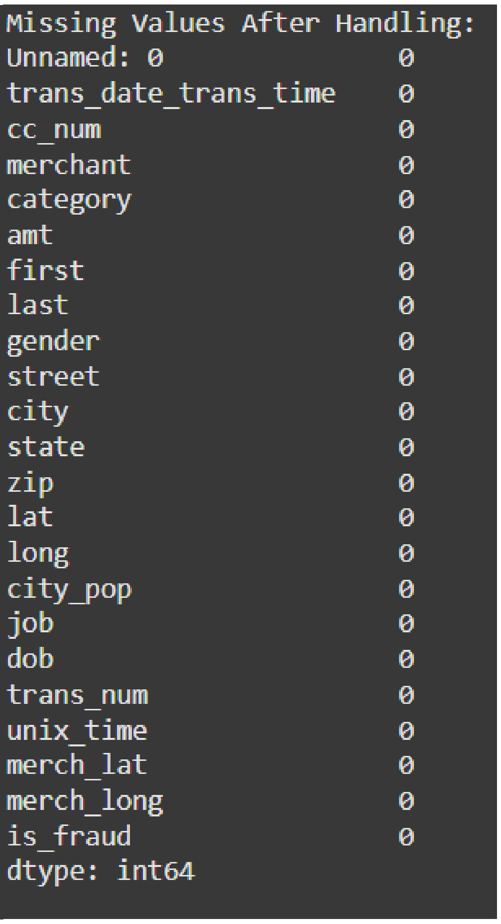
Part 2: Data Preprocessing

9. Handling Missing Data:

In the code, we handle missing data involves deciding how to treat missing values, either by removing them or imputing them.



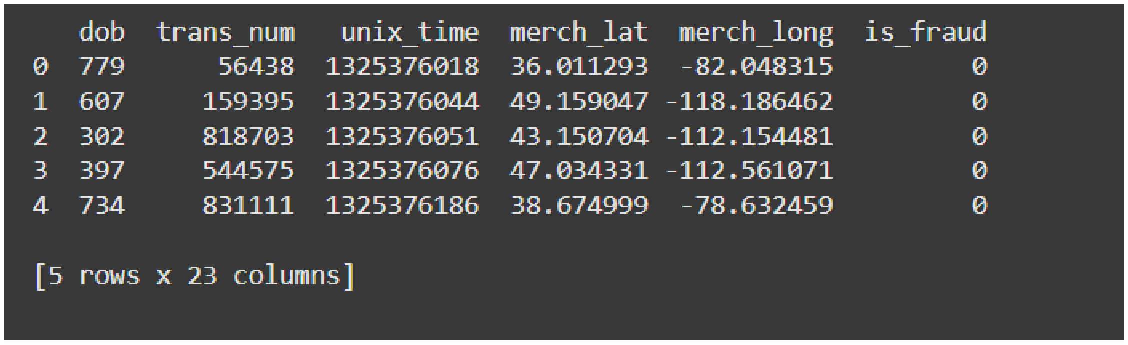
```
Missing Values in Each Column:
Unnamed: 0                    0
trans_date_trans_time         0
cc_num                        0
merchant                      0
category                      0
amt                           0
first                         0
last                          0
gender                        0
street                        0
city                          0
state                         0
zip                           0
lat                           0
long                          0
city_pop                      0
job                           0
dob                           0
trans_num                     0
unix_time                     0
merch_lat                     0
merch_long                    0
is_fraud                      0
merch_zipcode            195973
dtype: int64
```



```
Missing Values After Handling:
Unnamed: 0               0
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
```

10. Encoding Categorical Variables:

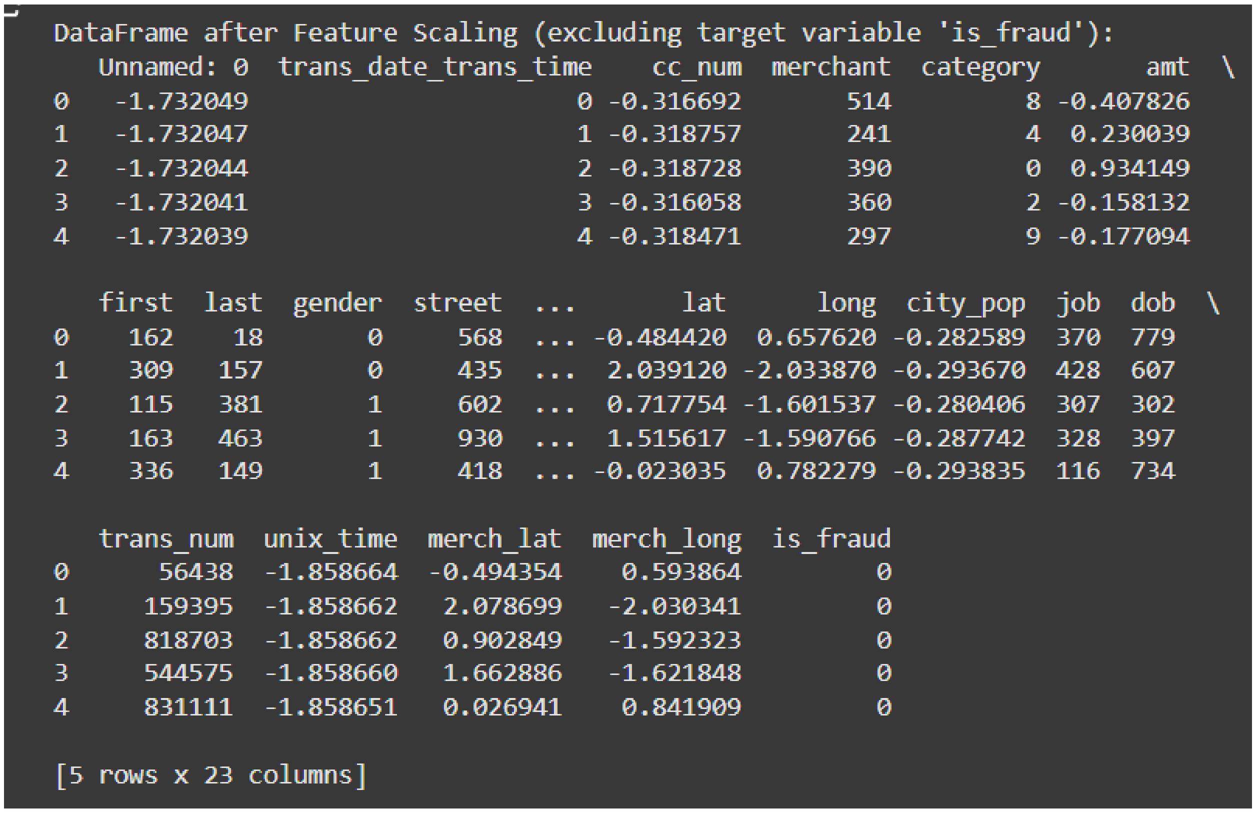
Categorical data needs to be encoded into numerical values for machine learning algorithms. Here, I'll use one-hot encoding for categorical variables.



```
   dob  trans_num   unix_time  merch_lat  merch_long  is_fraud
0  779      56438  1325376018  36.011293  -82.048315         0
1  607     159395  1325376044  49.159047 -118.186462         0
2  302     818703  1325376051  43.150704 -112.154481         0
3  397     544575  1325376076  47.034331 -112.561071         0
4  734     831111  1325376186  38.674999  -78.632459         0

[5 rows x 23 columns]
```

11. Feature Scaling:

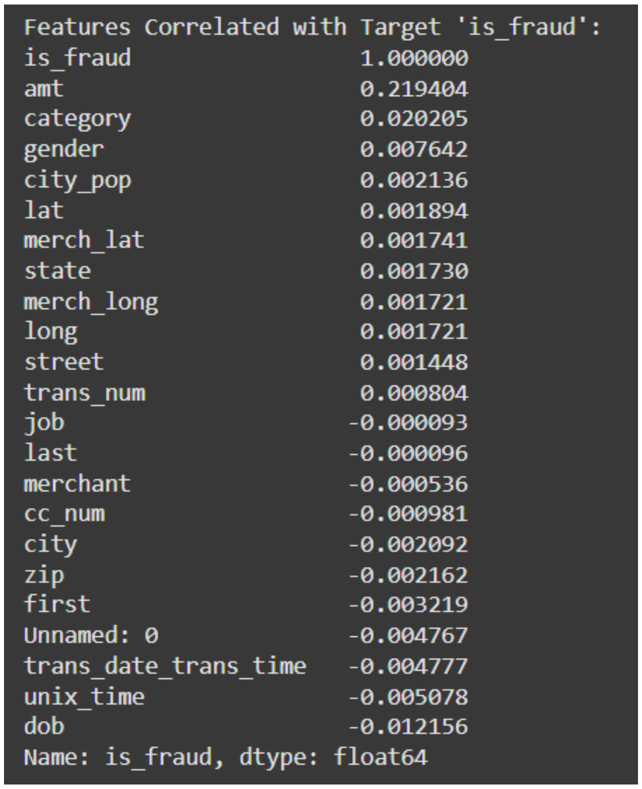
Feature scaling ensures that all numerical features have the same scale, which is important for distance-based algorithms.
We use standard scaling on the code.



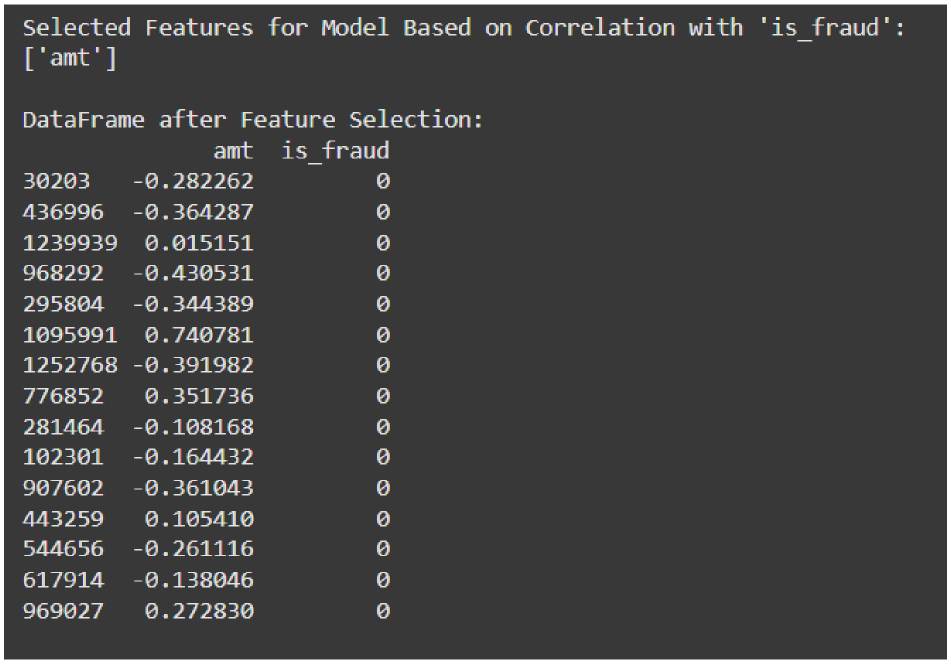
```
DataFrame after Feature Scaling (excluding target variable 'is_fraud'):
   Unnamed: 0  trans_date_trans_time    cc_num  merchant  category       amt  \
0   -1.732049                      0 -0.316692       514         8 -0.407826
1   -1.732047                      1 -0.318757       241         4  0.230039
2   -1.732044                      2 -0.318728       390         0  0.934149
3   -1.732041                      3 -0.316058       360         2 -0.158132
4   -1.732039                      4 -0.318471       297         9 -0.177094

   first  last  gender  street  ...       lat      long  city_pop  job  dob  \
0    162    18       0     568  ... -0.484420  0.657620 -0.282589  370  779
1    309   157       0     435  ...  2.039120 -2.033870 -0.293670  428  607
2    115   381       1     602  ...  0.717754 -1.601537 -0.280406  307  302
3    163   463       1     930  ...  1.515617 -1.590766 -0.287742  328  397
4    336   149       1     418  ... -0.023035  0.782279 -0.293835  116  734

   trans_num  unix_time  merch_lat  merch_long  is_fraud
0      56438  -1.858664  -0.494354    0.593864         0
1     159395  -1.858662   2.078699   -2.030341         0
2     818703  -1.858662   0.902849   -1.592323         0
3     544575  -1.858660   1.662886   -1.621848         0
4     831111  -1.858651   0.026941    0.841909         0

[5 rows x 23 columns]
```

12. . Feature Selection:

Feature scaling ensures that all numerical features have the same scale, which is important for distance-based algorithms. I'll use standard scaling here.



```
Features Correlated with Target 'is_fraud':
is_fraud                 1.000000
amt                      0.219404
category                 0.020205
gender                   0.007642
city_pop                 0.002136
lat                      0.001894
merch_lat                0.001741
state                    0.001730
merch_long               0.001721
long                     0.001721
street                   0.001448
trans_num                0.000804
job                     -0.000093
last                    -0.000096
merchant                -0.000536
cc_num                  -0.000981
city                    -0.002092
zip                     -0.002162
first                   -0.003219
Unnamed: 0              -0.004767
trans_date_trans_time   -0.004777
unix_time               -0.005078
dob                     -0.012156
Name: is_fraud, dtype: float64
```



```
Selected Features for Model Based on Correlation with 'is_fraud':
['amt']

DataFrame after Feature Selection:
              amt  is_fraud
30203   -0.282262         0
436996  -0.364287         0
1239939  0.015151         0
968292  -0.430531         0
295804  -0.344389         0
1095991  0.740781         0
1252768 -0.391982         0
776852   0.351736         0
281464  -0.108168         0
102301  -0.164432         0
907602  -0.361043         0
443259   0.105410         0
544656  -0.261116         0
617914  -0.138046         0
969027   0.272830         0
```

Part 3: Data Modeling

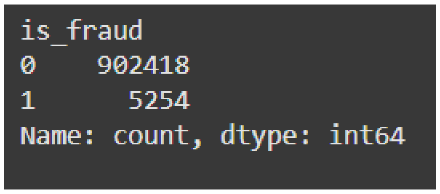
13. Algorithm Selection

Algorithm Selection For analyzing the Credit Card Transactions dataset, the problem type is typically binary classification (fraud detection) or unsupervised clustering (customer segmentation). Classification: Logistic Regression and Decision Trees: Good for interpretability and fast training. Random Forest and XGBoost: Effective for handling imbalanced datasets common in fraud detection. Neural Networks: Suitable for large datasets with complex patterns but require careful tuning. Clustering: K-Means: Useful for grouping similar transaction patterns but sensitive to initial clustering. DBSCAN: Handles varying densities, useful for anomaly detection in high-dimensional data.

XGBoost is the best choice for fraud detection due to its high accuracy, handling of imbalanced data, and ability to deal with complex patterns in large datasets. Here's a brief comparison with other algorithms: Logistic Regression: Why not: It assumes a linear relationship between the features and the target variable, which may not capture the complex patterns in fraud detection. It's also less effective for imbalanced datasets. Decision Trees: Why not: While interpretable, decision trees tend to overfit easily, especially with complex datasets like fraud detection. They struggle to generalize well without pruning or ensemble methods. Random Forest: Why not: Random Forest is more robust than decision trees and handles overfitting better. However, XGBoost often outperforms it in terms of both speed and accuracy, especially when tuned properly. Neural Networks: Why not: While effective, neural networks require large datasets and extensive tuning. They also lack interpretability, which is crucial in fraud detection. XGBoost is faster and more interpretable.

14. Data Splitting

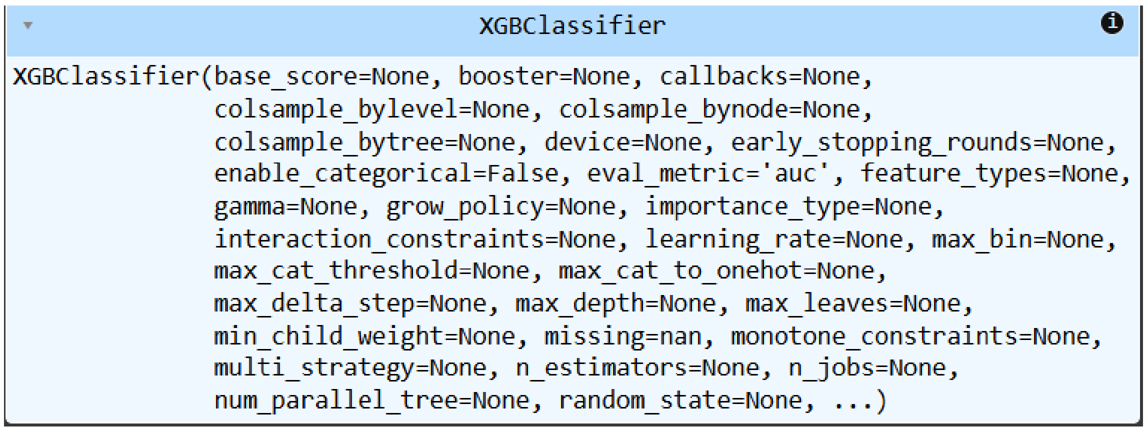
We'll use the train-test split method to divide the dataset into training and testing sets. We will also stratify the split to ensure that the class distribution (fraud vs. non-fraud) remains balanced across both sets.



```
is_fraud
0    902418
1      5254
Name: count, dtype: int64
```

15. Model Training

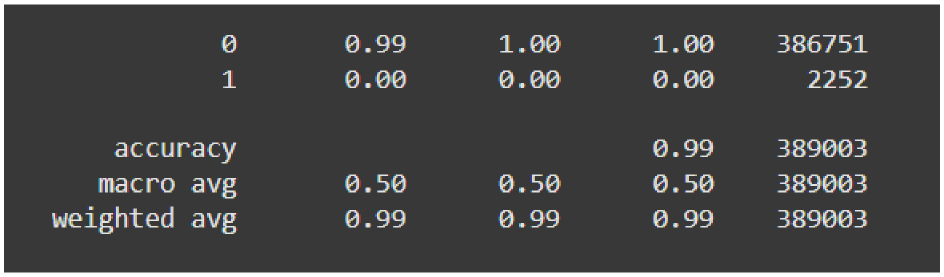
We'll train an XGBoost classifier. XGBoost is configured with the binary logistic regression objective for fraud detection and uses AUC (Area Under the Curve) as the evaluation metric.



```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='auc', feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

16. Model Evaluation

To evaluate the performance of the model, we will use a variety of metrics: Confusion Matrix, Classification Report, and AUC-ROC score. These will give us insights into the model’s ability to detect fraud (True Positives) and avoid false alarms (False Positives).



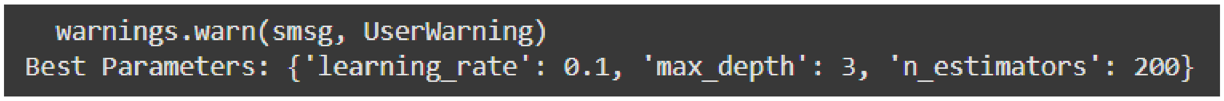
```
           0       0.99      1.00      1.00    386751
           1       0.00      0.00      0.00      2252

    accuracy                           0.99    389003
   macro avg       0.50      0.50      0.50    389003
weighted avg       0.99      0.99      0.99    389003
```

17. Performance Analysis

In this step, we assess how well the model performs based on the evaluation metrics: Confusion Matrix: This helps us understand how many fraud transactions (True Positives) and legitimate transactions (True Negatives) the model correctly identified, as well as the false positives and false negatives. Classification Report: It provides a detailed view of Precision, Recall, and F1-Score, which are crucial for imbalanced datasets. AUC-ROC: AUC measures the model's ability to distinguish between classes. A higher AUC indicates a better model.

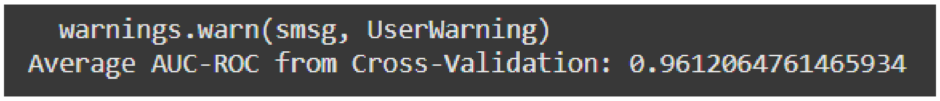
18. Model Improvement

To improve the model, we can use techniques like hyperparameter tuning, feature engineering, or trying different algorithms. Below is an example of tuning hyperparameters using GridSearchCV.



```
  warnings.warn(smsg, UserWarning)
Best Parameters: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200}
```

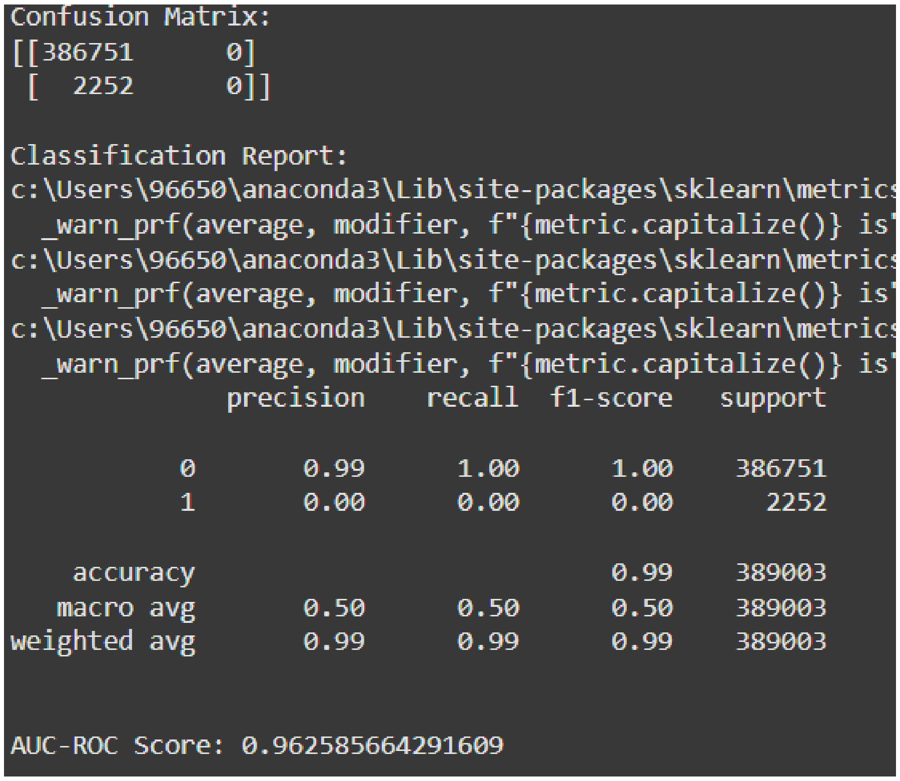
19. Validation To ensure that the model generalizes well, we can use k-fold cross-validation. This divides the training data into multiple subsets and trains the model on different combinations of training and validation data.



```
  warnings.warn(smsg, UserWarning)
Average AUC-ROC from Cross-Validation: 0.9612064761465934
```

20. Final Model Selection

After testing different hyperparameters and models, we will choose the model that provides the best performance based on the AUC-ROC, precision, and recall values. XGBoost, after hyperparameter tuning, tends to perform well in terms of accuracy and generalization on imbalanced datasets.



```
Confusion Matrix:
[[386751      0]
 [  2252      0]]

Classification Report:
c:\Users\96650\anaconda3\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is
c:\Users\96650\anaconda3\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is
c:\Users\96650\anaconda3\Lib\site-packages\sklearn\metrics
  _warn_prf(average, modifier, f"{metric.capitalize()} is
              precision    recall  f1-score   support

           0       0.99      1.00      1.00    386751
           1       0.00      0.00      0.00      2252

    accuracy                           0.99    389003
   macro avg       0.50      0.50      0.50    389003
weighted avg       0.99      0.99      0.99    389003


AUC-ROC Score: 0.962585664291609
```

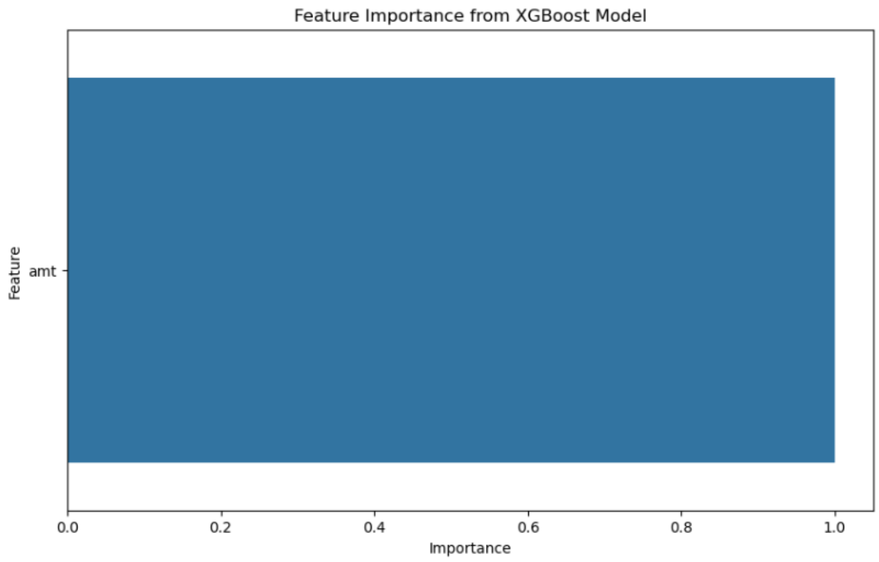
Data visualization

21. Data Distribution

We will visualize how different features are distributed using various plots, including histograms, boxplots, and bar plots for categorical features. This helps in identifying patterns, outliers, and anomalies.

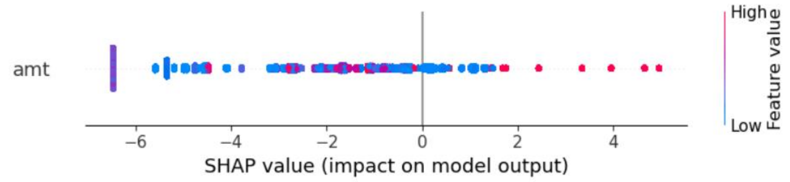
22. Feature Importance

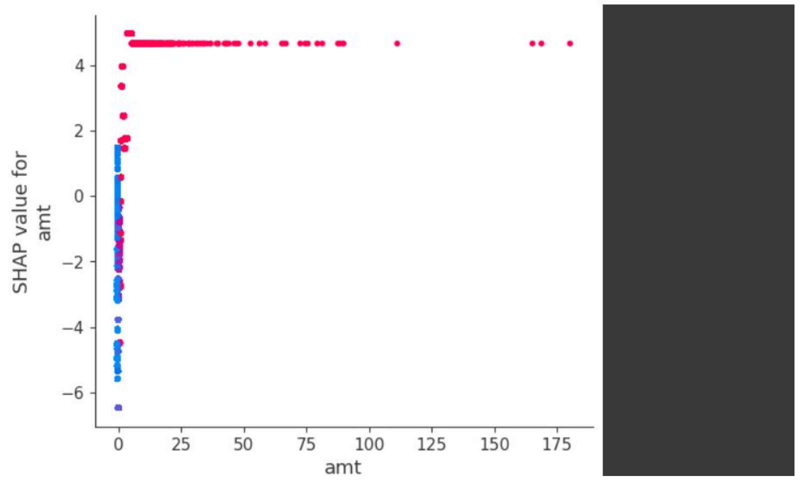
Feature importance will be visualized using the XGBoost model. Tree-based models like XGBoost provide a built-in way to interpret which features are most influential.



23. Model Performance Across Features

We will explore how the model’s predictions change across different values of features using Partial Dependence Plots (PDP) and SHAP values for more advanced interpretability.


amt
SHAP value (impact on model output)
High
Low
Feature value


SHAP value for amt
amt