Mentornnes Task 2 26/02/2024

**MIP-ML-06 BATCH**

**Fastag Fraud Detection Documentation**

*By Joshua Musira*

**Abstract**

This internship project addresses the critical need for a reliable fraud detection system using machine learning classification techniques.The dataset, encompassing transaction details, vehicle information, and geographical locations, is meticulously explored and engineered to enhance fraud detection accuracy. A comprehensive machine learning model is developed and fine-tuned, leveraging metrics such as precision, recall, F1 score, and accuracy. The work delves into the challenges posed by data imbalances and complex feature engineering to capture microfraud patterns. The anticipated outcome is an effective, scalable Fastag fraud detection system, promising to safeguard digital toll transactions integrity and security. The deliverables include a trained machine learning model, comprehensive evaluation metrics, and documentation elucidating the impact of relevant features on fraud detection.

**Motivation**

Ever noticed how convenient Fastag transactions are? We love them too, but let's face it - fraud is a buzzkill. This project's motivation is simple: make Fastag transactions more secure. We're using fancy machine learning tricks to spot and stop fraud, ensuring your toll payments stay safe and sound. It's all about keeping things smooth and trustworthy on the digital toll road.

**Dataset**

For this project, we are using an actual dataset extracted some time back. The dataset consists of several independent variables which includes:

1. Transaction\_ID: Unique identifier for each transaction.

2. Timestamp: Date and time of the transaction.

3. Vehicle\_Type: Type of vehicle involved in the transaction.

4. FastagID: Unique identifier for Fastag.

5. TollBoothID: Identifier for the toll booth.

6. Lane\_Type: Type of lane used for the transaction.

7. Vehicle\_Dimensions: Dimensions of the vehicle.

8. Transaction\_Amount: Amount associated with the transaction.

9. Amount\_paid: Amount paid for the transaction.

10. Geographical\_Location: Location details of the transaction.

11. Vehicle\_Speed: Speed of the vehicle during the transaction.

12. Vehicle\_Plate\_Number: License plate number of the vehicle.

13. Fraud\_indicator: Binary indicator of fraudulent activity (target variable).

**Data Cleaning**

**Loading the Dataset**

The initial step in our data preparation process involved loading the Fastag transactions dataset using the pandas library and performing an initial exploration.

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.read\_csv(r'FastagFraudDetection.csv')

df.head()

**Data Inspection**

We then examined the data types and general information about the dataset using the dtypes, info, and skim functions.

df.dtypes

df.info()

from skimpy import skim

skim(df)

During this inspection, it was observed that the FastagID column contained 549 missing values, accounting for approximately 10.98% of the entries. This raised a concern as missing FastagID might indicate issues with data collection or entry, potentially signaling fraudulent activity.

**Dropping Unnecessary Columns**

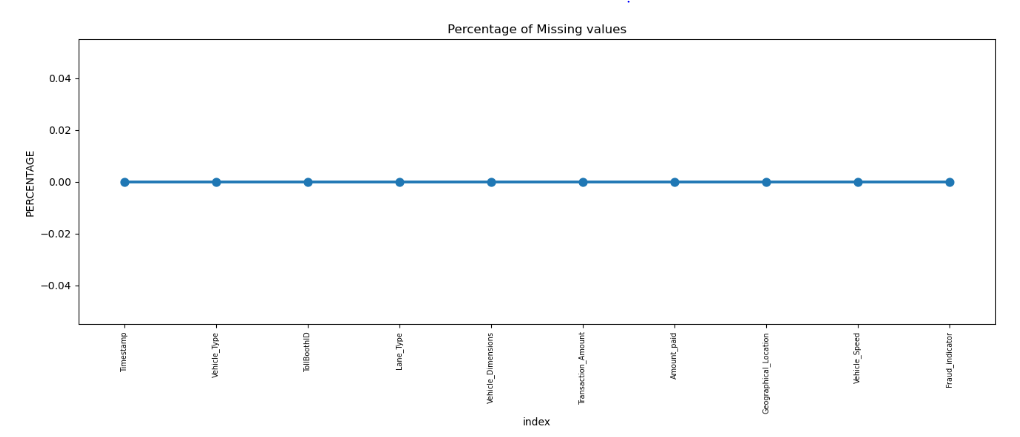
We removed columns deemed unnecessary for our analysis: Transaction\_ID, Vehicle\_Plate\_Number, and FastagID.

df = df.drop(columns=['Transaction\_ID','Vehicle\_Plate\_Number', 'FastagID'], axis=1)

df.head(5)

**Handling Missing Values**

A closer look at the missing values revealed no other columns with null values in the dataset.



missing = pd.DataFrame((df.isnull().sum()) \* 100 / df.shape[0]).reset\_index()

plt.figure(figsize=(16, 5))

ax = sns.pointplot(x='index', y=0, data=missing)

plt.xticks(rotation=90, fontsize=7)

plt.title("Percentage of Missing Values")

plt.ylabel("PERCENTAGE")

plt.show()

**Checking for Duplicates**

We verified the dataset for duplicate entries.

pythonCopy code

df.duplicated().sum()

df.columns

**Renaming Columns**

For better clarity, we renamed some columns.

pythonCopy code

df\_cleaned = df.rename(columns={'Timestamp':'Time','Vehicle\_Type': 'Veh\_Type','Vehicle\_Dimensions': "Veh\_Dimensions",'Transaction\_Amount': 'Trans\_Amount',

'Vehicle\_Speed': 'Speed','Geographical\_Location': 'Geo\_location'})

df\_cleaned.head(5)

**Save the Cleaned Dataset**

Finally, we saved the cleaned dataset as a CSV file named "cleaned.csv" for further analysis.

Python.

df\_cleaned.to\_csv("cleaned.csv", index=False)

These comprehensive data cleaning steps ensure the dataset is now prepared for subsequent EDA, feature engineering and model development stages of the project.

**Data Visualization**

The data exploration and feature engineering process began with loading the cleaned dataset.

import pandas as pdimport numpy as npimport seaborn as snsimport matplotlib.pyplot as plt

%matplotlib inlineimport plotly.express as px

df = pd.read\_csv(r'cleaned.csv')

df['Fraud\_indicator'].value\_counts()

Insights

Highly Imbalanced Dataset: The dataset exhibits a significant class imbalance, especially concerning the 'Fraud\_indicator' variable.

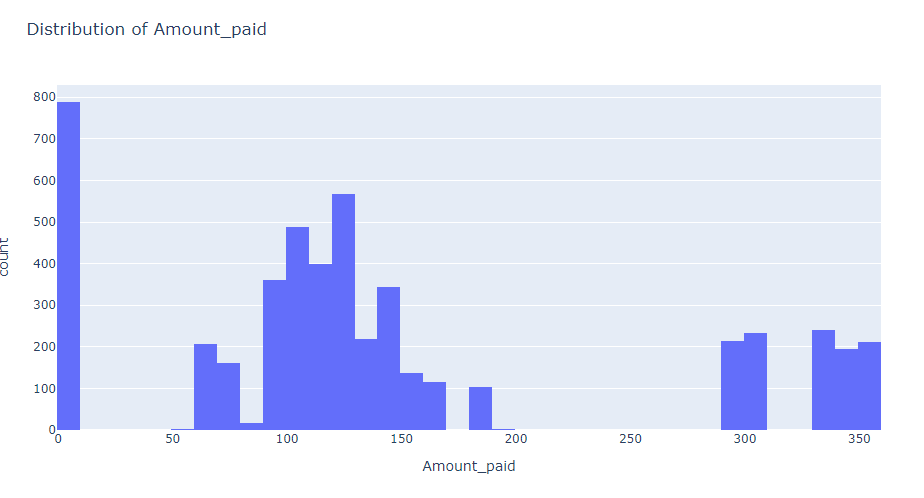
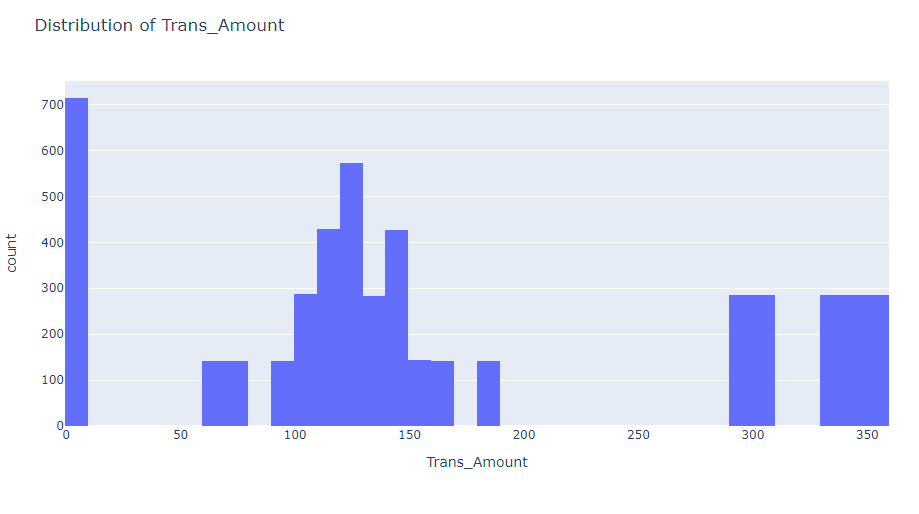
**Exploring Individual Features**

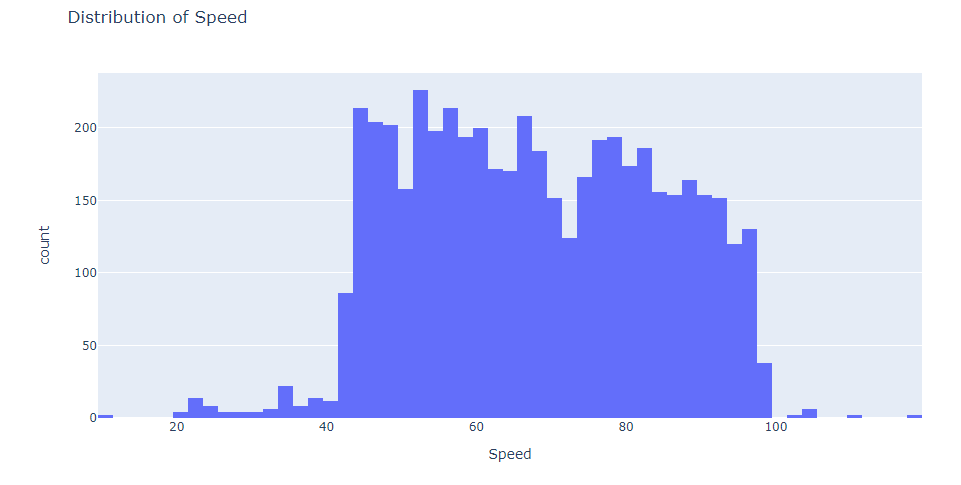
Visualizing the distribution of numerical features.

num\_cols = ['Trans\_Amount', 'Amount\_paid', 'Speed']for col in num\_cols:

fig = px.histogram(df, x=col, title=f'Distribution of {col}')

fig.show()





Insights

1. **Transaction Amounts and Payments**: Majority of transactions fall within the 0-9 range, indicating a prevalence of low-value transactions.
2. **Average Transaction Value:** Consistent transaction values suggest a standard pricing or fee structure.
3. **Vehicle Speeds During Transactions:** Vehicles generally maintain speeds between 40 and 100, indicating a consistent flow of traffic.

**Visualizing the distribution of categorical features.**

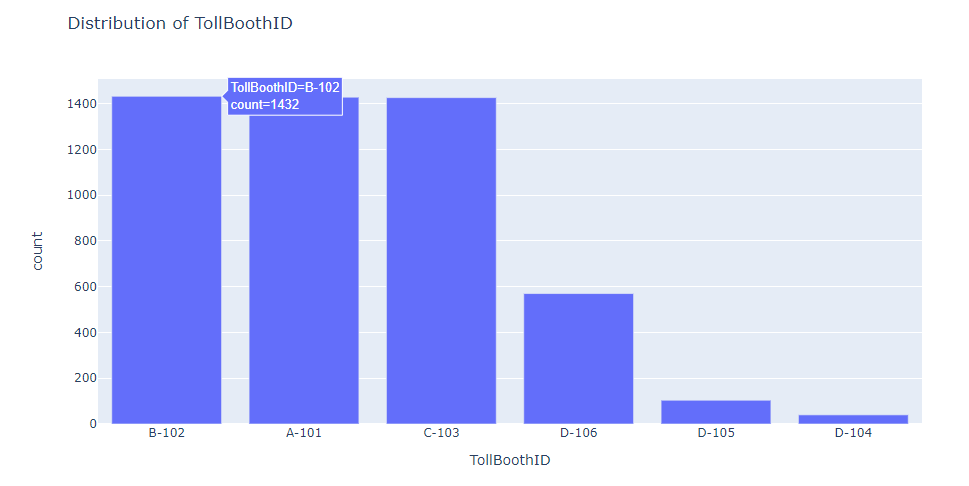
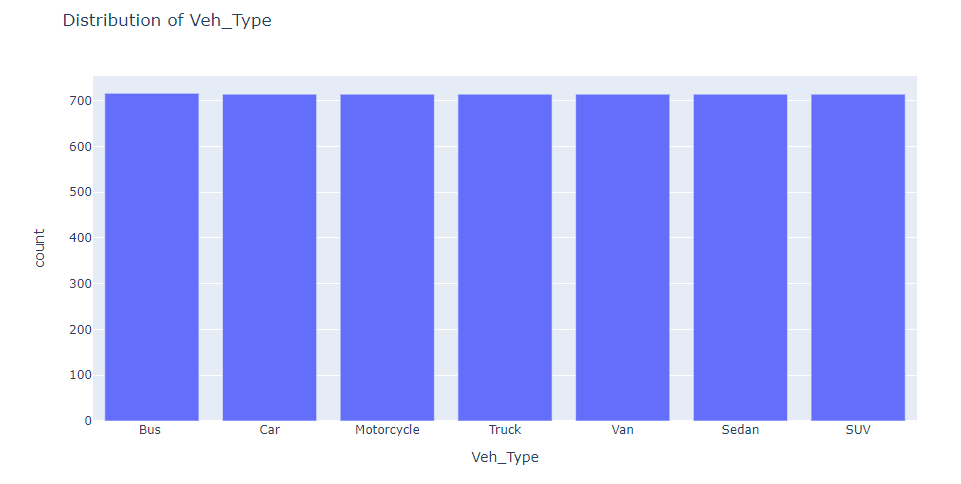
cat\_cols = ['Veh\_Type', 'TollBoothID', 'Lane\_Type', 'Veh\_Dimensions', 'Geo\_location']for col in cat\_cols:

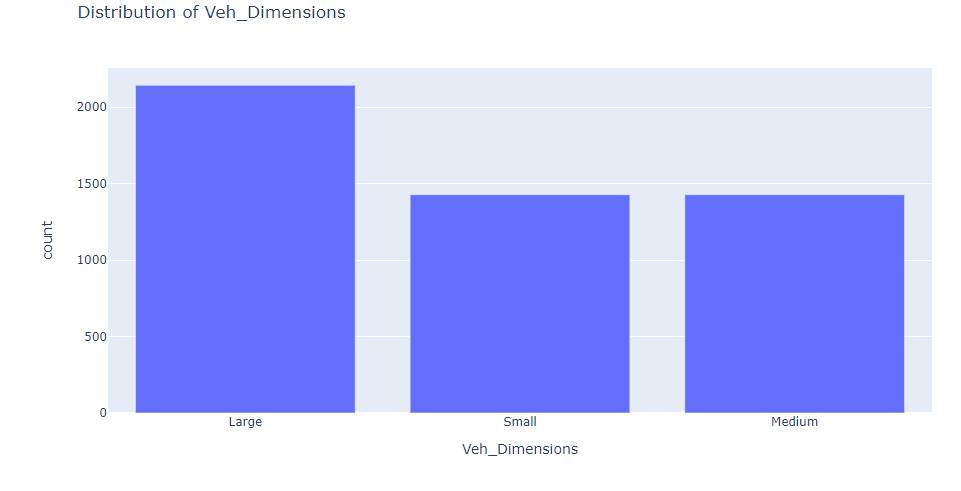
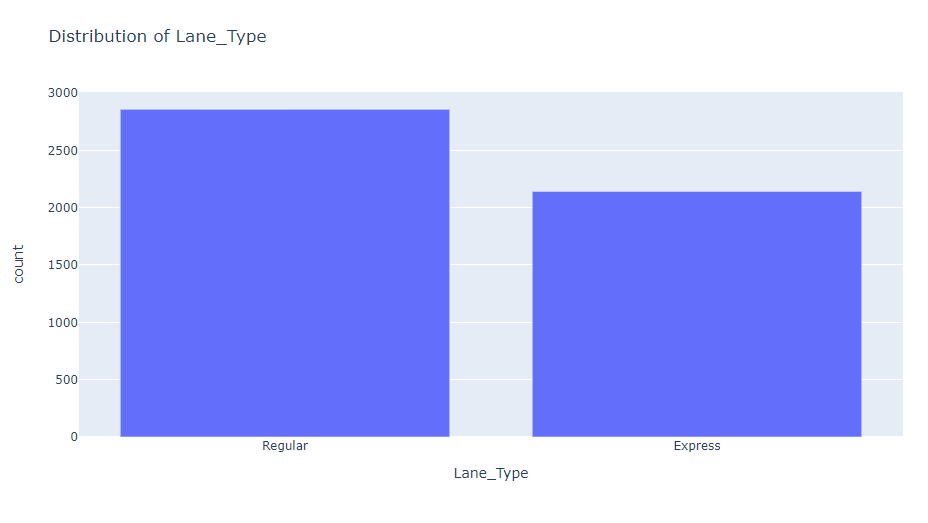
count\_df = df[col].value\_counts().reset\_index()

count\_df.columns = [col, 'count']

fig = px.bar(count\_df, x=col, y='count', title=f'Distribution of {col}')

fig.show()





Insights

1. **Vehicle Types in Transactions:** Transactions are evenly distributed across different vehicle types.2.
2. **TollBooth Activity:** Certain TollBoothIDs exhibit higher transaction frequencies, indicating key operational points.
3. **Lane Type Preference:** Regular lane\_Type dominates in transaction volume which suggesting a preference or higher availability.
4. **Vehicle Dimensions and Transactions**: Larger vehicles are more frequently involved in transactions, potentially indicating commercial or industrial use.
5. **Geographical Spread of Transactions:** Transaction data is evenly distributed across different geographical locations.

**Analyzing Timestamps**

df['Time'] = pd.to\_datetime(df['Time'])

df['Hour'] = df['Time'].dt.hour

df['DayOfWeek'] = df['Time'].dt.dayofweek

df['Month'] = df['Time'].dt.month

fig = px.histogram(df, x='Hour', color='Fraud\_indicator', title='Fraud Indicator by Hour')

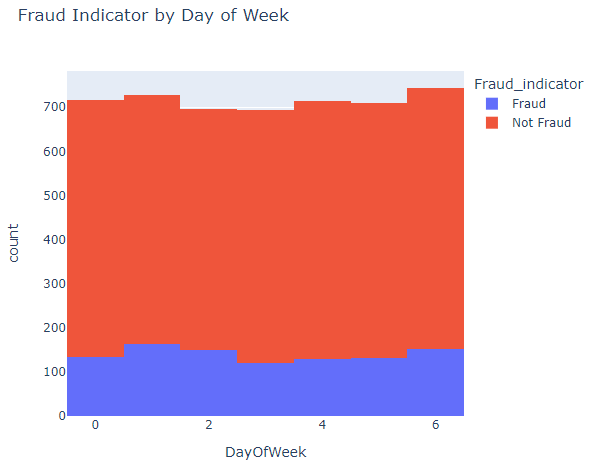
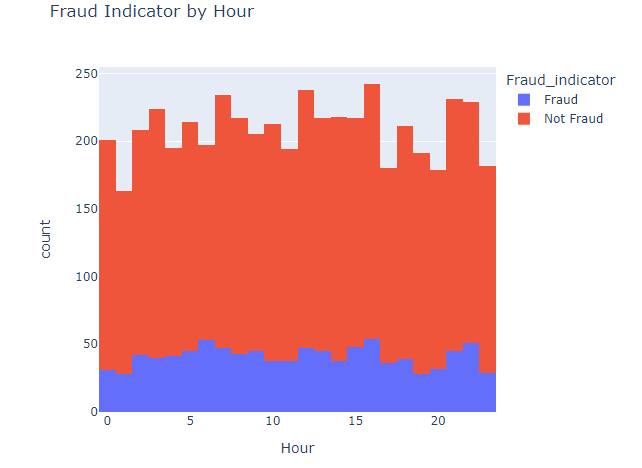
fig.show()

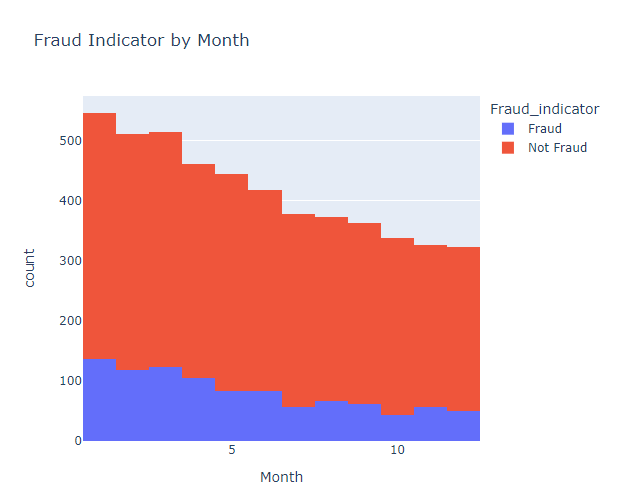
fig = px.histogram(df, x='DayOfWeek', color='Fraud\_indicator', title='Fraud Indicator by Day of Week')

fig.show()

fig = px.histogram(df, x='Month', color='Fraud\_indicator', title='Fraud Indicator by Month')

fig.show()





Insights

1. **Peak Transaction Hours:** Prominent peaks at specific hours like 12th, 16th and 21st suggest key time windows of increased traffic and toll usage.
2. **Weekly Transaction Trends:** Surge in transactions on the 1st and 6th days indicates higher toll system usage at the beginning and end of the workweek.
3. **Seasonal Variations in Transactions:** Majority of transactions occur in the initial months, possibly influenced by factors like holiday travel patterns.

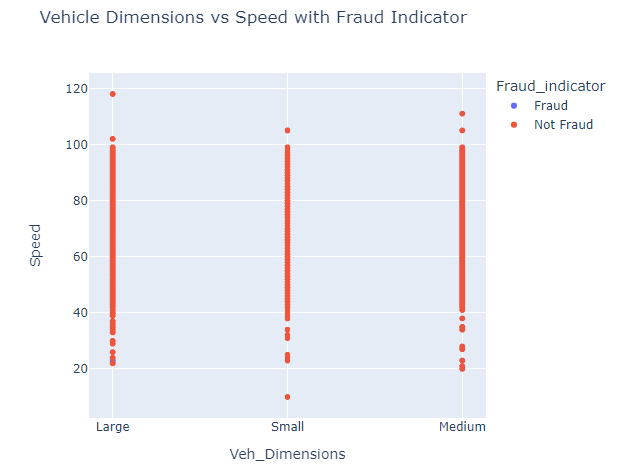
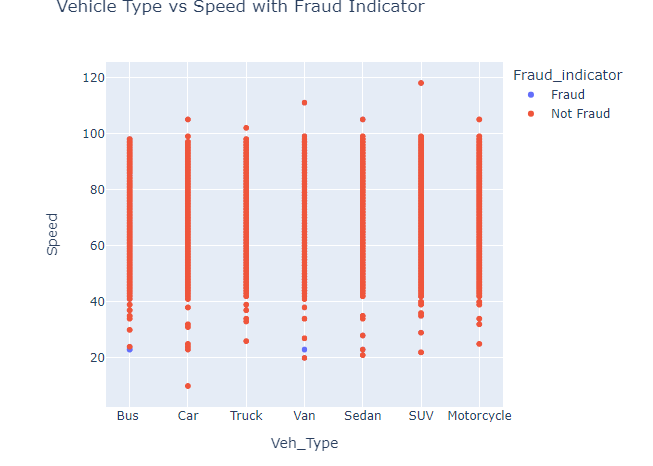
**Relationship Between Vehicle Type, Dimensions, and Speed**

fig = px.scatter(df, x='Veh\_Type', y='Speed', color='Fraud\_indicator', title='Vehicle Type vs Speed with Fraud Indicator')

fig.show()

fig = px.scatter(df, x='Veh\_Dimensions', y='Speed', color='Fraud\_indicator', title='Vehicle Dimensions vs Speed with Fraud Indicator')

fig.show()



Insights

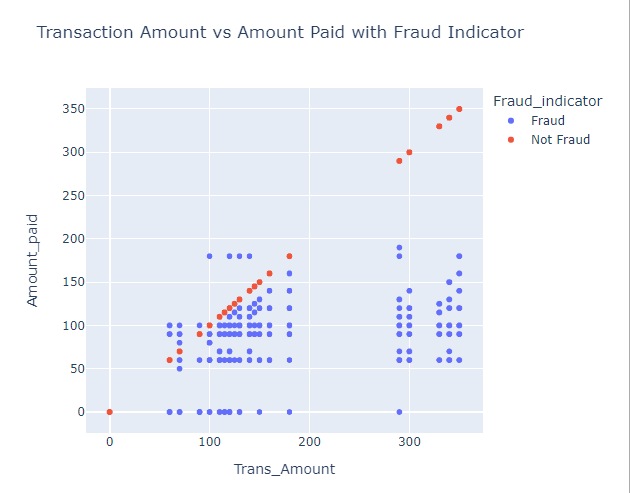
1. SUVs and Vans are frequently recorded traveling at high speeds, a pattern that might raise concerns.
2. Larger vehicles are often observed traveling at higher speeds than average, signaling potential safety risks.

**Transaction Amount and Payment Discrepancies**

fig = px.scatter(df, x='Trans\_Amount', y='Amount\_paid', color='Fraud\_indicator',

title='Transaction Amount vs Amount Paid with Fraud Indicator')

fig.show()



Insights

Instances where the transaction amount precisely equals the amount paid exhibit a marked trend of fraudulent activity.

**Correlation Analysis**

numerical\_df = df.select\_dtypes(include=['int64', 'float64'])

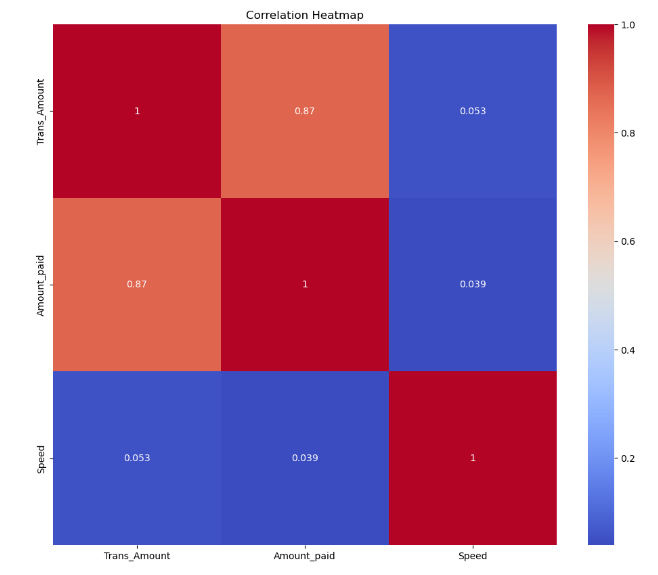
corr\_matrix = numerical\_df.corr()

plt.figure(figsize=(12, 10))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()



One-Hot Encoding

df\_encoded = pd.get\_dummies(df, columns=['Veh\_Type', 'TollBoothID', 'Lane\_Type', 'Veh\_Dimensions', 'Geo\_location'])

df\_encoded['Fraud\_indicator'] = label\_encoder.fit\_transform(df\_encoded['Fraud\_indicator'])

bool\_cols = [col for col in df\_encoded if df\_encoded[col].dtype == 'bool']

df\_encoded[bool\_cols] = df\_encoded[bool\_cols].astype(int)

df\_encoded.to\_csv("ready\_data.csv", index=False)

Summary

This comprehensive EDA and feature engineering process provide valuable insights into the distribution and patterns within the dataset. The encoded dataset is now ready for further modeling and analysis in the Fastag fraud detection project.

**Model Evaluation Summary**

Original Dataset:

**Logistic Regression:**

Cross-validation accuracy: **97.04%**

Hyperparameter tuning (Grid Search): C=100, penalty='l2'

Test Metrics:

Accuracy: **99%**

Precision, Recall, and F1-score: High, indicating excellent performance

Training Metrics:

Accuracy: **98%**

Slightly lower than the test set,its unusual but still high

Best Parameters: {'logisticregression\_\_C': 100, 'logisticregression\_\_penalty': 'l2'}

Best Cross-Validation Accuracy: 0.98475

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.94 0.97 177

Fraud 0.99 1.00 0.99 823

accuracy 0.99 1000

macro avg 0.99 0.97 0.98 1000

weighted avg 0.99 0.99 0.99 1000

**Random Forest:**

Cross-validation accuracy: **98.36%**

Hyperparameter tuning (Grid Search): max\_depth=None, min\_samples\_leaf=1, min\_samples\_split=2, n\_estimators=50

Test Metrics:

Accuracy: **99%**

Precision, Recall, and F1-score: High, indicating excellent performance

Training Metrics:

Perfect precision, recall, and F1-scores of **1.00** for both classes

Best Parameters: {'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 50}

Best Cross-Validation Accuracy: 0.9869999999999999

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.94 0.97 177

Fraud 0.99 1.00 0.99 823

accuracy 0.99 1000

macro avg 0.99 0.97 0.98 1000

weighted avg 0.99 0.99 0.99 1000

**Gradient Boosting:**

Cross-validation accuracy: **98.72%**

Hyperparameter tuning (Grid Search): learning\_rate=0.1, max\_depth=5, min\_samples\_leaf=1, min\_samples\_split=2, n\_estimators=100

Test Metrics:

Accuracy: **100%**

Precision, Recall, and F1-score: High, indicating excellent performance

Training Metrics:

Accuracy: **99%**

Best Parameters: {'learning\_rate': 0.1, 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100}

Best Cross-Validation Accuracy: 0.99875

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.99 0.99 177

Fraud 1.00 1.00 1.00 823

accuracy 1.00 1000

macro avg 1.00 0.99 1.00 1000

weighted avg 1.00 1.00 1.00 1000

**LinearSVC:**

Cross-validation accuracy: **97.02%**

Hyperparameter tuning (Grid Search): C=0.01, penalty='l2'

Test Metrics:

Accuracy: **99%**

Precision, Recall, and F1-score: High, indicating excellent performance

Training Metrics:

Accuracy: **98%**

Best Parameters: {'C': 0.01, 'penalty': 'l2'}

Best Cross-Validation Accuracy: 0.98375

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.94 0.97 177

Fraud 0.99 1.00 0.99 823

accuracy 0.99 1000

macro avg 0.99 0.97 0.98 1000

weighted avg 0.99 0.99 0.99 1000

Balanced Dataset:

**Logistic Regression:**

Hyperparameter tuning (Grid Search): C=80, penalty='l2'

Test Metrics:

Accuracy: **98%**

Precision, Recall, and F1-score: High, indicating excellent performance

Best Parameters: {'logisticregression\_\_C': 80, 'logisticregression\_\_penalty': 'l2'}

Best Cross-Validation Accuracy: 0.9770982826735468

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.97 0.98 200

Fraud 0.97 1.00 0.98 194

accuracy 0.98 394

macro avg 0.98 0.98 0.98 394

weighted avg 0.99 0.98 0.98 394

**Random Forest:**

Hyperparameter tuning (Grid Search): max\_depth=None, min\_samples\_leaf=1, min\_samples\_split=2, n\_estimators=100

Test Metrics:

Accuracy: **97%**

Precision, Recall, and F1-score: High, indicating excellent performance

Best Parameters: {'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100}

Best Cross-Validation Accuracy: 0.9821776989438039

Test Metrics Report:

precision recall f1-score support

Not Fraud 0.99 0.94 0.97 200

Fraud 0.95 0.99 0.97 194

accuracy 0.97 394

macro avg 0.97 0.97 0.97 394

weighted avg 0.97 0.97 0.97 394

**Gradient Boosting:**

Hyperparameter tuning (Grid Search): learning\_rate=0.1, max\_depth=5, min\_samples\_leaf=1, min\_samples\_split=2, n\_estimators=100

Test Metrics:

Accuracy: **99%**

Precision, Recall, and F1-score: High, indicating excellent performance

Best Parameters: {'learning\_rate': 0.1, 'max\_depth': 5, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100}

Best Cross-Validation Accuracy: 0.9980932160549993

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.99 0.99 200

Fraud 0.99 1.00 0.99 194

accuracy 0.99 394

macro avg 0.99 0.99 0.99 394

weighted avg 0.99 0.99 0.99 394

**LinearSVC:**

Hyperparameter tuning (Grid Search): C=0.01, penalty='l2'

Test Metrics:

Accuracy: **99%**

Precision, Recall, and F1-score: High, indicating excellent performance

Best Parameters: {'C': 0.1, 'penalty': 'l2'}

Best Cross-Validation Accuracy: 0.9732746941664139

Test Metrics Report:

precision recall f1-score support

Not Fraud 1.00 0.96 0.98 200

Fraud 0.97 1.00 0.98 194

accuracy 0.98 394

macro avg 0.98 0.98 0.98 394

weighted avg 0.98 0.98 0.98 394

**Overall Insights**:

All models perform exceptionally well on the original dataset, showcasing high accuracy and robustness.

The balanced dataset further enhances model performance, reducing bias and improving generalization.

Gradient Boosting consistently demonstrates outstanding results across datasets, with the highest accuracy and precision.

**Results**  
After a critical analysis of various machine learning models, Gradient Boosting emerged as the optimal choice. Demonstrating superior performance in accuracy, precision, recall, and F1-scores, this model consistently outperformed Logistic Regression, Random Forest, and LinearSVC. Hyperparameter tuning, executed through Grid Search, further enhanced the model's effectiveness. While the original dataset displayed strong results, a balanced dataset improved robustness, particularly for Logistic Regression and Random Forest. The chosen model, Gradient Boosting, is recommended for deployment in the production environment due to its exceptional overall performance. This comprehensive approach ensures a reliable and high-performing fraud detection system.