Fraud Detection Using Logistic Regression

1. How did you clean the data (missing values, outliers, multicollinearity)?

I started by checking for missing values using df.isnull().sum() and confirmed there weren't any significant missing values in the dataset.

For outliers, I used box plots to visually inspect all numerical features. This gave me a good idea of where unusual values might be, but I didn't remove any at this stage to avoid dropping potentially useful fraud data.

To handle multicollinearity, I calculated the Variance Inflation Factor (VIF) for each numeric column (excluding the target variables). This helped identify any highly correlated features, which can distort the model.

2. Can you explain your fraud detection model?

I used Logistic Regression for this project. It's a linear classification model that works well for binary outcomes like fraud or not fraud.

Here's how it works: the model takes a linear combination of the input features and passes it through a sigmoid function to output a probability between 0 and 1. If the probability is above a certain threshold (usually 0.5), the transaction is predicted as fraudulent.

I chose this model because it's simple, fast to train, and easy to interpret. That's important when explaining results to non-technical stakeholders. Of course, for more complex fraud patterns, other models like Random Forest or XGBoost might perform better - but Logistic Regression is a great starting point.

3. How did you select the features for your model?

I used a few approaches to decide which features to keep:

- VIF (Variance Inflation Factor) to check multicollinearity and avoid redundant features.
- Correlation analysis with the target variable (isFraud) to see which features had the strongest relationship with fraud.
- I also removed irrelevant columns like nameOrig and nameDest since they're identifiers and don't help in prediction.

- Lastly, I applied one-hot encoding to the type column to handle the categorical data properly.

4. How did you evaluate the model's performance?

I used a few standard evaluation metrics:

- Confusion Matrix to get a basic idea of how many true/false positives and negatives the model predicted.
- Classification Report to see precision, recall, and F1-score.
- ROC-AUC Score to measure how well the model separates fraud from non-fraud.

The results showed that the model performs reasonably well, especially given the simplicity of Logistic Regression. It provides a solid baseline to compare future models.

5. What are the key factors that predict fraud in your model?

Based on the model coefficients (sorted by their absolute value), the most important features were:

- newbalanceOrig (strong negative): A lower balance after the transaction is often linked to fraud - which makes sense, as fraudulent transactions might drain the account.
- oldbalanceOrg (positive): Accounts with higher starting balances seemed more likely to be involved in fraud.
- amount (negative): Interestingly, larger transaction amounts were associated with lower fraud risk in this dataset. That was a bit surprising.
- type_PAYMENT (negative): This suggests that regular payments are less likely to be fraudulent, which aligns with what we'd expect.

6. Do these factors make sense? Why or why not?

Some do, some don't - here's my reasoning:

- newbalanceOrig (-) makes a lot of sense. Fraud often leaves the sender's account empty.
- oldbalanceOrg (+) could make sense too - attackers might go after accounts with more money.
- amount (-) doesn't match our usual assumptions. We'd expect higher amounts to be riskier. This could be due to the dataset's specific nature or limitations of the linear model.
- type_PAYMENT (-) makes sense. Payments are common and usually legit.

Since this is a Logistic Regression model (which is linear), it might not capture all the complex relationships.

Also, these are correlations - they don't prove causation.

7. What kind of fraud prevention should a company focus on while updating its systems?

Here's what I'd recommend for any company improving its fraud detection system:

- Real-time transaction monitoring to catch fraud before it completes.
- Stronger authentication, like multi-factor authentication (MFA), especially for high-value transactions.
- Encrypted data storage and transmission to protect customer info.
- Anomaly detection systems that learn normal user behavior and flag suspicious patterns.
- Device fingerprinting and IP tracking to spot unusual access.
- Transaction limits - both in frequency and amount - to block rapid attacks.
- User education on phishing and account security.
- Regular audits and patching to keep infrastructure secure.
- Simple fraud reporting tools so users can report suspicious activity quickly.

8. How would you measure whether those security measures are working?

To know if the prevention steps are effective, I'd track the following:

- Fraud rates over time - if they drop after implementation, that's a good sign.
- False positives and negatives - we don't want to block too many legit users.
- User feedback - are users frustrated with new steps, or do they feel safer?
- A/B testing - try changes on a small group and compare results.
- Response time - are we catching fraud fast enough?
- Security audits - to ensure systems are working as expected and not vulnerable.
- Behavior changes - track if users are adapting well to new systems.

If needed, I can also share the full code or visualizations used during the project.

Thanks for taking the time to read through this!

- Sudha Kumari

(Data Science & Fraud Detection Enthusiast)