TASK_INSAID

1. Data cleaning including missing values, outliers and multi-collinearity.

In [46]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Load the dataset
df = pd.read_csv("C:/Users/DELL/Downloads/Fraud.csv")
```

In [47]:

```
df.head()
```

Out[47]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703

In [48]:

```

columns_to_drop = ['nameOrig','nameDest']
df.drop(columns_to_drop, axis=1, inplace=True)
```

In [51]:

```python
# Check for missing values
print('Missing values in the dataset:')
print(df.isnull().sum())
```

```
Missing values in the dataset:
step              0
type              0
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

In [53]:

```python
# Remove missing values
df.dropna(inplace=True)
```

In [54]:

```python
# Check for outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
df_out = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
outlier_count = len(df) - len(df_out)
print(f'Number of outliers removed: {outlier_count}')
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_42796\840714081.py:5: FutureWar
ning: Automatic reindexing on DataFrame vs Series comparisons is deprecate
d and will raise ValueError in a future version. Do `left, right = left.al
ign(right, axis=1, copy=False)` before e.g. `left == right`
  df_out = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axi
s=1)]

Number of outliers removed: 2043214
```

In [56]:

```python
# Check for multicollinearity
# Scale the features using StandardScaler
scaler = StandardScaler()
X = df.drop('type', axis=1)
y = df['type']
X_scaled = scaler.fit_transform(X)
```

In [57]:

```
# Calculate the VIF for each feature
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X_scaled, i) for i in range(X_scaled.shap
vif["features"] = X.columns
print(vif)
```

```
   VIF Factor        features
0    1.003137            step
1    3.771634          amount
2  502.913267   oldbalanceOrg
3  504.282321  newbalanceOrig
4   66.101079  oldbalanceDest
5   76.200749  newbalanceDest
6    1.186855         isFraud
7    1.002562  isFlaggedFraud
```

In [61]:

```
# Drop variables with VIF greater than 5
vif_variables = vif[vif['VIF Factor'] > 5]['features'].tolist()
df = df.drop(vif_variables, axis=1)
```

In [62]:

```
# Save the cleaned dataset
df_out.to_csv('cleaned__dataset.csv', index=False)
```

3. How did you select variables to be included in the model?

In [107]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['type'] = le.fit_transform(df['type'])
```

In [108]:

```
from sklearn.linear_model import Lasso

X = df.drop('type', axis=1)
y = df['type']

model = Lasso(alpha=0.1)
model.fit(X, y)

print(model.coef_)
```

```
[-1.13370396e-05 -3.60771297e-06  4.15768548e-06 -6.20471271e-06
 -6.73326873e-07  1.36698137e-07  0.00000000e+00  0.00000000e+00]
```

2. Describe your fraud detection model in elaboration.

In [69]:

```
df = pd.read_csv("C:/Users/DELL/cleaned__dataset.csv")
```

In [70]:

```
X = df.drop('type', axis=1)
y = df['type']
```

In [96]:

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Fit logistic regression model
lr = LogisticRegression()
lr.fit(X_train, y_train)

# Predict on test data
y_pred = lr.predict(X_test)
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

4. Demonstrate the performance of the model by using best set of tools.

In [103]:

```
# Import libraries
from sklearn.metrics import confusion_matrix, accuracy_score
# Import necessary libraries
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc


# Evaluate the performance of the model using various metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred,average='weighted')

# Print the results
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 0.9426877744877193
Precision: 0.9025135087133099
Recall: 0.9426877744877193
F1 Score: 0.9192927668020416
```

In [104]:

```
# Get feature coefficients
coef_df = pd.DataFrame(lr.coef_[0], index=X.columns, columns=['Coefficient'])
coef_df = coef_df.abs().sort_values(by='Coefficient', ascending=False)
print(coef_df)
```

```
                Coefficient
newbalanceOrig     0.000729
oldbalanceOrg      0.000721
newbalanceDest     0.000240
amount             0.000220
oldbalanceDest     0.000100
step               0.000015
isFraud            0.000000
isFlaggedFraud     0.000000
```

5. What are the key factors that predict fraudulent customer?

High transaction amounts, High frequency of transactions, High amount of transactions per hour, High transaction amount per hour.

6. Do these factors make sense? If yes, How? If not, How not?

These factors make sense as they are indicators of abnormal behavior in terms of the volume and frequency of transactions. Fraudulent customers are likely to make large and frequent transactions in a short period of time, which is different from normal customers. By identifying these factors, the company can take proactive measures to prevent fraud, such as implementing transaction monitoring or flagging suspicious transactions for manual review.

7.What kind of prevention should be adopted while company update its infrastructure?

While updating the infrastructure, the company can take the following prevention measures to mitigate the risk of fraud:

Regularly monitor and update the security systems to ensure they are up-to-date and effective. Implement two-factor authentication to verify the identity of the user during login. Conduct regular security training for employees to make them aware of the latest fraud schemes and how to prevent them. Implement transaction monitoring to identify unusual or suspicious activity. Use machine learning algorithms to detect anomalies in transaction patterns and identify potential fraud.

8. Assuming these actions have been implemented, how would you determine if they work?

To determine if the implemented actions are effective in preventing fraud,

we can measure the performance of the model on a separate holdout dataset. We can calculate various evaluation metrics such as precision, recall, accuracy, and F1-score on this dataset to evaluate the model's performance. If the performance of the model is significantly improved after implementing these actions, it can be concluded that these actions are effective in preventing fraud. Additionally, the company can monitor the frequency and type of fraudulent activities over time and compare them before and after implementing these actions to evaluate their effectiveness.

In [ ]:

```

```