1. Best Data Cleaning Techniques:

Based on our dataset and implementation,
top data cleaning techniques used:

1. Handling Missing Values: None in our case, but typically replaced with mean/median/imputed values.

2. Data Normalization: Min-Max Scaler used to scale numerical columns between 0 and 1.

3. Encoding Categorical Variables: Label Encoding used for 'type' column and Category Codes for 'nameOrig' and 'nameDest'.

4. Removing Duplicates: Not necessary in our dataset, but typically done using drop_duplicates().

5. Outlier Detection: Not explicitly done, but some algorithms (like SVM) are robust to outliers.

2. Best Model and Why: Best Model: XGBoost

Why:

1. HIGHEST ACCURACY SCORE:

   XGBoost surpasses all other models with the highest accuracy score (0.999828).

2. CONSISTENT PERFORMANCE:

   Ensemble methods like XGBoost tend to perform consistently well across various datasets.

3. ROBUST AGAINST OVERFITTING:

   XGBoost's regularization techniques help reduce overfitting risk.

3. Final Model with Prediction:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC


# Load data in chunks
chunk_size = 100000  # Adjust based on your system's capacity
chunks = pd.read_csv(r"C:\Users\91703\Downloads\Online fraud detection.csv",
chunksize=chunk_size)

# Example: Process chunks
for chunk in chunks:
    # Perform operations on each chunk
    print(chunk.head())


#Normalize numerical columns
numeric_cols = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
chunk[numeric_cols] = chunk[numeric_cols].apply(lambda x: (x - x.min()) /
(x.max() - x.min()))



#Encode categorical columns
chunk['type'] = chunk['type'].map({'CASH_OUT': 0, 'CASH_IN': 1, 'DEBIT': 2,
'PAYMENT': 3, 'TRANSFER': 4})
chunk['nameOrig'] = chunk['nameOrig'].astype('category').cat.codes
chunk['nameDest'] = chunk['nameDest'].astype('category').cat.codes
```

```
#Split data into features (X) and target variable (y)
X = chunk.drop('isFraud', axis=1)
y = chunk['isFraud']


#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)

#Scale data using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


   #  Naive Bayes classifier #
nb_model = GaussianNB()
#Train Naive Bayes model
nb_model.fit(X_train, y_train)



#Make predictions on test set
y_pred_nb = nb_model.predict(X_test)
print(y_pred_nb)



#Evaluate model performance
print("Naive Bayes Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_nb))
print("Classification Report:", classification_report(y_test, y_pred_nb))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred_nb))
import warnings
warnings.filterwarnings('ignore')
```

Naive Bayes Model Performance:
Accuracy: 0.9871332990221308
Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	9590
1	0.00	0.00	0.00	125
accuracy			0.99	9715
macro avg	0.49	0.50	0.50	9715
weighted avg	0.97	0.99	0.98	9715

Confusion Matrix: [[9590 0]
[125 0]]