Credit Card Fraud Detection(Development Part-2)

Detecting credit card fraud using data science typically involves building a machine learning model to identify potentially fraudulent transactions. Here's a high-level guide to developing such a system, along with some code examples in Python. Please note that this is a simplified example, and real-world credit card fraud detection systems are much more complex and use extensive datasets.

1. **Data Collection:**

Gather a dataset of credit card transactions. You can use public datasets like Kaggle's Credit Card Fraud Detection dataset.

Python code

```
Import pandas as pd

Df = pd.read_csv("creditcard.csv")
```

2. **Data Preprocessing**:

Explore and clean the data. Check for missing values and outliers.

Normalize numerical features, like 'Amount' and 'Time'.

```
From sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()

Df['normalized_Amount'] = scaler.fit_transform(df['Amount'].values.reshape(-1, 1))
```

3. **Feature Selection:**

Select relevant features for the model. In this case, all features except 'Time' and 'Amount' may be relevant.

```
X = df.drop(['Time', 'Amount', 'Class'], axis=1)

Y = df['Class']
```

4. **Split Data:**

Split the data into training and testing sets.

Python code

```
From sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. **Model Selection:**

Choose an appropriate machine learning algorithm for classification. Random Forest and Gradient Boosting are often used for this task.

Python code

```
From sklearn.ensemble import RandomForestClassifier


Model = RandomForestClassifier()
```

6. **Model Training:**

Train the model on the training data.

Python

Copy code

```
Model.fit(X_train, y_train)
```

7. **Model Evaluation:**

Evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and ROC AUC.

Python code

```
Print(confusi From sklearn.metrics import classification_report, confusion_matrix


Y_pred = model.predict(X_test)

Print(classification_report(y_test, y_pred))
```

on_matrix(y_test, y_pred))

8. **Hyperparameter Tuning:**

Optimize the model's hyperparameters to improve performance.

9. **Deployment**:

Once satisfied with the model's performance, deploy it in a production environment where it can continuously monitor incoming transactions for fraud.

Please note that a real-world system would also include ongoing monitoring, retraining, and integration with a transaction processing system. Additionally, dealing with imbalanced datasets (few fraud cases compared to non-fraud) requires advanced techniques like oversampling, undersampling, or using anomaly detection algorithms.

Done by:

ABHIRAM KB

Reg no:720921244001